CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

From Logs to Insights: Exploring User Behavior in RobotStudio

Master's thesis in Computer science and engineering

JINXUAN LI
SHIJIA TANG

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

From Logs to Insights: Exploring User Behavior in RobotStudio

JINXUAN LI
SHIJIA TANG

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

JINXUAN LI
SHIJIA TANG

Supervisor: Carl-Johan Seger, Computer Science and Engineering
Supervisor: Henrik Jansson Valter, Computer Science and Engineering
Advisor: Marcus Flurr, ASEA Brown Boveri
Examiner: Johannes Åman Pohjola, Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

JINXUAN LI
SHIJIA TANG
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Understanding how users interact with software is essential—not only for designing intuitive interfaces but also for building meaningful, behavior-driven test scenarios. In this study, we explore user behavior in ABB RobotStudio by analyzing a large dataset of backend log files, each recording detailed event traces from real usage sessions.

To approach this problem from multiple angles, we employ three methods. First, we apply N-gram analysis to examine what users do, what events tend to occur together, and in what order, providing us with a window into common behavioral patterns. Second, we construct first-order Markov chains to model the likelihood of transitioning from one type of event to another, capturing dynamics user actions. Third, we use clustering and N-gram to investigate whether different types of event sequences naturally emerge. This helps us uncover whether there are distinct features and recurring patterns of usage across clusters.

Together, these three methods reveal both structural and sequential aspects of how users interact with RobotStudio. We find that certain behaviors repeat consistently across users and sessions, while others are more context-dependent. These insights offer practical value: they can support user-centered UI improvements and help generate test cases that more closely mirror real-world workflows.

Keywords: log analysis, Longformer, clustering, N-gram, test generation.

Acknowledgements

First, we would like to thank our academic supervisors, Carl and Henrik, who have been guiding us through our thesis. You both have been very supportive and have offered us continuous feedback to improve our project. We would also like to extend our appreciation to the people working at ABB Mölndal who have helped us collect the data. Without your help, our project would not have been completed on time. Special thanks to our industrial supervisor, Marcus from ABB, who helped us narrow down the topic and provided valuable suggestions. Last but not least, we would like to thank Johannes for serving as our examiner, taking the time to read our thesis, and providing us with valuable feedback.

Shijia Tang, Jinxuan Li, Gothenburg, 2025-06-04

Contents

List of Figures

List of Tables

1

Introduction

This chapter begins by presenting the background of the master's thesis, including a brief introduction to the case company and one of its current challenges. The following sections outline the thesis purpose, research questions, scope, and delimitations.

1.1 Background

In an era with rapid technological advancement, robotics has emerged as a key component of Industry 4.0, offering substantial capabilities in manufacturing [1].

This master's thesis was conducted in collaboration with Asea Brown Boveri (referred to hereafter as ABB). As one of the leading organizations in the industrial robotics market, ABB offers a broad portfolio of advanced robotic manipulators [2]. The revenue contribution of ABB Robotics has ranged between $2.25 billion and $2.75 billion [3]. This segment encompasses industrial robots, autonomous mobile robots (AMRs), robotic application cells, intelligent systems, and digital services. Approximately 70% of these sales are made directly to customers [3].

One of ABB's notable innovations is RobotStudio, a software suite widely recognized for its offline programming and simulation capabilities for robotic applications. Developed using ABB's virtual controller technology, RobotStudio allows users to design, simulate, and program ABB robots in a virtual environment. The software ensures that simulated robot movements accurately mirror real-world behavior [4], significantly reducing commissioning time and increasing productivity. Figure 1.1 illustrates the user interface of RobotStudio ([1]), including key panels such as the Top Menu, the Left Panel (displaying components and mechanisms), the 3D Viewport (showing the robot cell and motion paths), the Document Panel, and the Output Panel (reporting controller status and robot activity).

Despite its capabilities, further investigation is needed to gain a deeper understanding of RobotStudio users. Currently, innovation within RobotStudio is primarily driven by customer interviews, user feedback, competitive analysis, and expert intuition, etc. While these methods are valuable, they can be subjective [5], time-consuming [6], non-scalable [7], and insufficient in capturing implicit user needs [8].

On the other hand, RobotStudio event log files, which are generated during daily

[1]Version 24.4.11080.0

software usage, offer an objective data source that has not been fully leveraged. The utilization of these event logs aligns closely with ABB's research and development framework [9]. They offer potential for continuous improvement [10], [11], strategic product development [12], and new service model exploration [13], all grounded in actual user interactions with ABB technology [9]. Thus, a more structured analytical approach is expected to extract actionable insights from RobotStudio event logs and enhance the prioritization of software development.

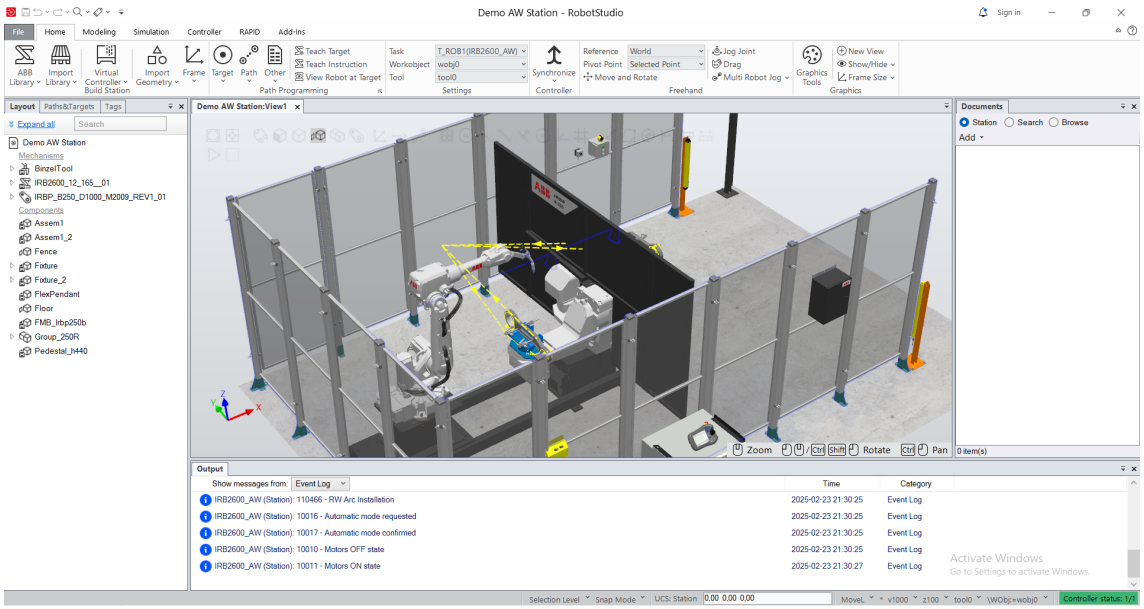

Figure 1.1: Example User Interface of RobotStudio (version 24.4.11080.0)

1.2 Purpose

The primary aim of this thesis is to explore user interaction patterns and contribute to more user-centric software solutions.

As robotic systems become increasingly complex and diverse in user applications, it is crucial for ABB to understand the distinct needs of various user groups. This project can identify different behavioral patterns by applying pattern mining techniques to user interaction data. Additionally, clustering based on these interactions may reveal unique user groups and their corresponding characteristics. These insights could inform the development of customized RobotStudio features and interfaces, ultimately improving user satisfaction.

With pattern mining and clustering algorithms, this research bridges the gap between log-based analytics and behavioral insight. The findings are intended to support user-centered development.

1.3 Research Questions

This thesis aims to address the following research questions:

- **RQ1**: Can meaningful user behavior patterns be identified from RobotStudio event logs?
- **RQ2**: Can the discovered user behavior patterns be utilized to generate realistic test cases?
- **RQ3**: Can distinct and meaningful user behavior patterns be discovered through the clustering of RobotStudio event logs?

1.4 Scope and Delimitations

1.4.1 Scope

This project focuses on analyzing user behavior through event log data collected from selected premium versions of RobotStudio. The objective is to identify interaction patterns among fee-paying users using clustering methodologies. The analysis is limited to quantitative user interaction data extracted from the log files; qualitative sources such as user interviews or surveys are excluded. Furthermore, the research focuses on log features that can be processed efficiently.

1.4.2 Delimitations

A key delimitation of this project is the nature of the dataset. The analysis relies solely on event log files that capture user interactions with RobotStudio. While these logs provide valuable behavioral indicators, they do not capture the all factors influencing user decisions. The exclusion of supplementary data, such as surveys or expert assessments, may constrain the depth of behavioral understanding.

Additionally, for pattern mining, the approaches such as n-grams and Markov chains have inherent limitations. These methods typically capture only short-term dependencies within fixed-length sequences and assume stationary patterns. This restricts their ability to accurately represent complex user behaviors, especially when long-term or irregular event relationships are involved.

Furthermore, only log files from recent RobotStudio versions are analyzed to maintain consistency and focus. This choice may limit the generalizability of findings across all versions and user groups.

Another limitation involves the trade-off between comprehensive feature extraction and the computational efficiency of clustering algorithms. Although the project aims to extract as many relevant features as possible, only the most significant features are retained to maintain algorithmic performance and ensure clear results.

Finally, the clustering process itself may introduce ambiguity when interpreting the resulting groups. Unclear or overlapping clusters can obscure meaningful behavioral patterns. Addressing such complexity may require exploring more advanced methods, such as recurrent neural networks, to capture nuanced user behaviors better.

2

Theory

This chapter begins by emphasizing the importance of analyzing user behavior for software development. It then continues by delving into the effect of log analysis in understanding user behavior, and test generation theory. Additionally, the chapter introduces commonly used natural language processing (NLP) approaches to log analysis, followed by the introduction to Sentence Transformers (SBERT). Finally, we discussed clustering, with an emphasis on its application to log analysis.

2.1 Importance of Analyzing User Behavior

Understanding user behavior is essential for effective product development. By analyzing how users interact with software, product managers and software developers can identify common bottlenecks [14], discover usability features [15] [16], detect emerging user needs, and finally enhance user experience. Rather than relying on assumptions or anecdotal feedback, this approach leads to better decision-making and enables development teams to prioritize tasks based on real-world user needs [17].

For software developers, user behavior features play a crucial role in planning development schedules. By identifying frequently encountered issues and most valued features, software engineers can allocate resources efficiently, optimize feature development, and minimize unnecessary iterations.

This data-driven approach ensures that development efforts align with user expectations and business goals, ultimately leading to more successful and user-centric software products [18].

2.2 Log Analysis in Understanding User Behaviors

System logs often contain redundant information, making it challenging to extract meaningful insights. Many studies have focused on using log events to analyze user behaviors by transforming raw log messages into structured representations that capture hidden patterns. This section provides a brief overview of research related to log-based user behavior mining.

Jeba et al. explored web server logs to understand customer browsing behavior [19]. Given the non-deterministic nature of user navigation, they employed soft clustering methods to assign membership values to sessions. A modified version of Fuzzy

C-Means clustering was used, incorporating access log preprocessing, user and session identification, and a Mountain Density Function (MDF)-based fuzzy clustering technique. The resulting clusters revealed common navigational behaviors.

Liu et al. introduced a two-layered framework to enhance behavior discovery in ProM by separately characterizing user interaction behavior and plug-in calling behavior [20]. They proposed detailed discovery techniques to construct a model of user interaction behavior.

Tao et al. investigated user behavior patterns by mining action data extracted from log files across different subsystems within their respective domains [21]. They identified distinct behavioral categories within various modules and subsystems of an intelligent manufacturing environment. Experimental results demonstrated that their approach effectively uncovered variations in behavioral aspects among cross-domain participants, particularly in resource access patterns, operational tasks, and performance assessments.

2.3 Pattern Mining

Pattern mining is a crucial data analysis technique used extensively for discovering frequent patterns or sequences in large datasets, such as logs from software applications and user interactions. Its primary goal is to uncover meaningful regularities, correlations, and structures that aid decision-making, optimize user experiences, and improve system efficiencies. Han et al. provided foundational concepts and techniques for pattern mining, highlighting its broad applicability in data-driven analyses [22].

In analyzing user behavior, n-gram analysis is widely utilized for detecting frequent sequential patterns of actions or events. Liu and Zhang introduced n-gram analysis in text mining contexts, emphasizing its effectiveness in capturing frequent sequential structures from sequential data, such as system logs or user interaction traces [23]. Xie and Phoha explored web logs using n-gram-based clustering to identify common user navigation patterns, demonstrating how this approach effectively highlights user behavior and anomalies [24].

Another prominent technique in pattern mining is the Markov Chain model, which captures user behaviors as probabilistic transitions among discrete states. Each state represents distinct user actions or system events. Deshpande and Karypis employed Markov models for web access prediction, illustrating their strength in probabilistically modeling user navigation sequences [25]. Chierichetti et al. questioned the Markovian nature of web user navigation by empirically validating the model's predictive power and its limitations in practical log analysis scenarios [26].

Combining n-gram analysis and Markov chain modeling forms a comprehensive analytical framework for interpreting complex log data, significantly enhancing the understanding of user behaviors in intricate software systems like RobotStudio.

2.4 Log Embedding Clustering

2.4.1 Log Clustering

In Section 2.2, we examined how logs can be used to analyze user behavior. A crucial technique that improves log analysis is log clustering. By grouping similar log entries, log clustering helps uncover user activity patterns, detect anomalies, and identify behavioral trends. It enables a more structured interpretation of the vast log data, making it easier to recognize recurring user actions and deviations from normal behavior. This method is crucial to improving the user experience, optimizing system performance, and enhancing security. This section provides an overview of key research efforts in log clustering.

Gurumdimma et al. developed an unsupervised algorithm called LogCluster to detect failure patterns in clustered logs by analyzing the sequential characteristics of log messages [27]. Their approach prioritizes message content for failure detection while using timestamps and node IDs only during preprocessing. The Generating Frequency Matrix process in LogCluster structures log messages to capture event distributions and uncover patterns. It extracts failure sequences, which consist of logs recorded within a specified time window (tw) before a failure event. These sequences are then converted into a term-frequency matrix (M), where rows represent unique message types and columns correspond to different failure sequences. Each column vector contains the frequency distribution of terms within a sequence, serving as input for clustering. This method facilitates effective pattern recognition, anomaly detection, and failure diagnosis. LogCluster is applied to three different log types, and the results are positive; runs that end in failure are detected with an average f-measure of 78%.

Vaarandi et al. introduced another LogCluster, a dynamic log clustering algorithm that identifies frequently occurring log patterns and outlier events [28]. LogCluster employs hierarchical clustering to extract log templates without predefined rules. It tokenizes log messages into static keywords and dynamic parameters and then groups similar logs using a similarity metric. New log entries are assigned to existing clusters or used to form new ones when no sufficiently similar cluster exists. The algorithm continuously updates clusters, enabling it to adapt to evolving log formats and making it effective for anomaly detection, log analysis, and large-scale system monitoring.

Thaler et al. proposed a log clustering method based on neural language models for the extraction of signatures in forensic logs [29]. Their approach, LSTM-AE+C, utilizes an RNN autoencoder to map log lines into an embedding space, where clustering is performed to assign signatures. Experiments on both proprietary and public datasets demonstrated that LSTM-AE+C outperforms two state-of-the-art methods in signature extraction.

Zhang et al. introduced LogRobust, a log-based anomaly detection approach that extracts the semantic information of log events and represents them as semantic vectors [30]. LogRobust utilizes an attention-based Bi-LSTM model to capture

contextual dependencies within log sequences and dynamically learn the significance of various log events. This enables the model to handle unstable log events and sequences effectively. Evaluated on logs from the Hadoop system and a Microsoft online service, LogRobust demonstrated strong performance in mitigating log instability and achieving accurate, robust anomaly detection in real-world, evolving log environments.

These studies illustrate different approaches to log clustering, ranging from traditional clustering algorithms to deep learning-based techniques. By effectively structuring log data, these methods enable improved system monitoring, enhanced anomaly detection, and more accurate behavior analysis.

2.4.2 The NLP approaches of Log Analysis

Log files, which consist of sequences of action descriptions accompanied by contextual information, exhibit structural and semantic similarities to natural language text. This has led to the widespread adoption of Natural Language Processing (NLP) techniques for analyzing log data. NLP techniques are particularly effective in tasks such as log parsing, anomaly detection, and pattern mining, where extracting meaningful features from unstructured log data is critical.

Early approaches to log analysis, such as those proposed by Bertero et al. [31], treated log files as conventional text and employed the word embedding technique Word2Vec. These methods map log entries into high-dimensional vector spaces, enabling the representation of log data in a format suitable for machine learning algorithms. However, traditional embedding techniques such as Word2Vec, GloVe, and Bag-of-Words (BoW) have a significant limitation: they generate static word representations that fail to capture the contextual nuances of log entries. This limitation hinders their ability to model the dynamic and context-dependent nature of log data effectively.

2.4.3 Introduction to Longformer

To address the issue discussed in Subsection 2.4.2, more advanced models, such as BERT (Bidirectional Encoder Representations from Transformers), have been introduced. BERT leverages a transformer-based architecture to capture bidirectional context, enabling it to generate context-aware embeddings that better represent the semantic meaning of log entries.

However, BERT models often face difficulties in capturing long-range dependencies in long texts due to self-attention limitations [32]. Longformer with an attention mechanism that scales linearly with the sequence Length is introduced to solve this problem, making it easy to process documents with about 16 thousand tokens.

For our task, we propose leveraging Longformer to generate semantic embeddings from RobotStudio event log files. These embeddings will capture both the contextual and sequential information inherent in the logs, providing a robust foundation for subsequent analysis. By applying clustering algorithms to these embeddings, we aim to identify meaningful patterns in user behavior that are both contextually rich and

computationally efficient. This approach not only enhances the interpretability of log data but also facilitates the discovery of actionable insights for improving user experience and system performance.

2.4.4 Clustering Techniques

Clustering techniques help categorize large volumes of log data by identifying similarities and patterns, potentially enabling more effective log analysis and interpretation. Different approaches offer unique advantages depending on the characteristics of the log data and the analysis objectives. In this chapter, we introduce some of the basic clustering methods commonly used in log analysis, highlighting their principles, strengths, and applications.

2.4.4.1 K-Means Clustering

K-Means is one of the most popular unsupervised learning algorithms that solves the well-known clustering problem. The aim of the K-Means algorithm is to divide N points in D-dimensions into K clusters so that the within-cluster sum of squares is minimized [33].

Let $X = \{x_1, x_2, ..., x_n\}$ be a dataset in the d-dimensional Euclidean space $\mathbb{R}^d$. Let $A = \{a_1, a_2, ..., a_k\}$ be the k cluster centers. Let the binary variable z_{ij} be the indicator showing whether the point x_i belongs to cluster j or not. The objective function of K-Means is $J(z, A) = \sum_{i=1}^{n} \sum_{j=1}^{k} z_{ij} ||x_i - a_j||^2$. The K-Means algorithm is iterated through necessary conditions for minimizing the K-Means objective function $J(z, A)$ with updating equations for cluster centers and memberships, respectively, as

$$a_k = \frac{\sum_{i=1}^{n} z_{ik} x_{ij}}{\sum_{i=1}^{n} z_{ik}} \quad \text{and} \quad z_{ik} = \begin{cases} 1, & \text{if } \|x_i - a_k\|^2 = \min\limits_{1 \le t \le c} \|x_i - a_t\|^2 \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

2.4.4.2 Hierarchical Clustering

Unlike the K-Means algorithm, which requires specifying the number of clusters a priori and relies on centroid-based optimization, hierarchical clustering operates without predefined cluster numbers and is guided by linkage criteria that determine how distances between clusters are measured. Hierarchical clustering yields a set of nested clusters, organized as a cluster tree known as a dendrogram. Each leaf node of the tree usually contains only one object, while each inner node represents the union of its children nodes [34].

The basic idea of hierarchical clustering algorithms is to construct a hierarchical relationship among data according to a dissimilarity or similarity measure between clusters [35].

Hierarchical clustering method is divided into two types: divisive and agglomerative [36]. The divisive hierarchical clustering method first sets all data points into one initial cluster, then divides the initial cluster into several sub-clusters, and

iteratively partitions these sub-clusters into smaller ones until each cluster contains only one data point or data points within each cluster are similar enough. In the agglomerative approach, each data point initially forms an individual cluster, and these clusters are iteratively merged based on minimum, maximum, average, or centroid linkage, ultimately resulting in a single cluster that encompasses all data points. The hierarchical structure is typically visualized using a dendrogram, which captures the merging or splitting process, enabling intuitive interpretation of cluster relationships and granularities.

2.4.4.3 Agglomerative Clustering

Agglomerative clustering is a bottom-up hierarchical clustering method in which each data point initially forms its own cluster. At each iteration, the algorithm merges the pair of clusters that are closest to each other based on a specified distance metric. This merging process continues until a single cluster encompassing all data points is formed. The outcome is a hierarchical structure of clusters, typically represented as a dendrogram, where any two clusters A and B at different levels of the hierarchy satisfy one of the following relationships: $A \cap B = \emptyset$, $A \subset B$, or $B \subset A$ [37]. This hierarchical structure is particularly useful in applications where the number of clusters is not known a priori or when the inheritance relationships among clusters are of interest, such as in certain bioinformatics analyses [37].

2.4.4.4 Spectral Clustering

Spectral clustering has emerged as an effective alternative to traditional clustering algorithms, particularly in situations where the data structure is not easily captured by convex shapes or linear boundaries [38]. Rather than operating directly in the input space, this method begins by constructing a similarity graph, where each data point is treated as a node and edges encode pairwise similarity, often computed using a radial basis function or cosine metric.

Once the similarity matrix is built, the algorithm computes the graph Laplacian and uses its eigenvectors to embed the data into a lower-dimensional space. Clustering is then performed in this spectral space, typically using a method such as K-Means. This approach allows the algorithm to capture the intrinsic geometry of the dataset, making it particularly useful for problems involving complex or nonlinear cluster boundaries.

Due to its ability to uncover latent structures in data, spectral clustering has been applied in various domains, including image segmentation, community detection, and user behavior modeling.

2.4.4.5 Gaussian Mixture Model Clustering

Gaussian Mixture Models (GMM) offer a flexible, probabilistic framework for clustering that assumes data points are generated from a mixture of multiple Gaussian distributions [39]. Each component in the mixture represents a cluster, characterized by its own mean and covariance. Rather than assigning points to clusters determin-

istically, GMM assigns probabilities, acknowledging uncertainty and allowing for overlapping regions between clusters.

The standard approach to fitting a GMM is the Expectation-Maximization (EM) algorithm, which iteratively estimates both the parameters of each Gaussian and the posterior probabilities of point membership. Compared to simpler methods like K-Means, GMM can model elliptical clusters and adapt to varying cluster densities and orientations, making it more suitable for real-world data that rarely conforms to ideal shapes.

In the context of software log analysis or test behavior modeling, GMMs can be particularly useful for discovering nuanced behavioral clusters, such as rare but significant usage paths, or distinguishing between typical and anomalous execution flows. Its probabilistic interpretation also supports uncertainty-aware test case prioritization and risk-sensitive testing strategies.

2.4.4.6 Clustering Evaluation Metrics

The following clustering evaluation metrics were selected to find the most suitable clustering algorithm for our log context clustering task:

- **Silhouette Score:**
  It checks how well each log fits in its cluster compared to other clusters. A higher score indicates that the logs in the same cluster are very similar to each other and distinct from those in other clusters. The Sihouette Score for a single data point is defined as [40]:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{2.2}$$

  $a(i)$ is the average intra-cluster distance (mean distance between i and all other points in the same cluster). $b(i)$ is the lowest average inter-cluster distance (mean distance between i and all points in the nearest cluster). The overall Silhouette Score is the mean of the Silhouette Score across all data points.

- **Davies-Bouldin Score:**
  This score measures the closeness of the clusters. A lower score indicates that clusters are tightly packed and far apart from each other. The Davies-Bouldin index is calculated as [40]:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} \left(\frac{S_i + S_j}{M_{ij}} \right) \tag{2.3}$$

  k is the number of clusters. S_i and S_j are the average distances of all points in clusters i and j to their respective centroids. M_{ij} is the Euclidean distance between the centroids of clusters i and j.

- **Calinski-Harabasz Score:**
  This compares how far apart clusters are from each other versus how spread

out the points are within each cluster. A higher score means better-defined clusters. The Calinski-Harabasz Score is defined as [40]:

$$CH = \frac{\mathrm{Tr}(B_k)}{\mathrm{Tr}(W_k)} \cdot \frac{n-k}{k-1} \tag{2.4}$$

$\mathrm{Tr}(B_k)$ is the trace of the between-cluster dispersion matrix. $\mathrm{Tr}(W_k)$ is the trace of the within-cluster dispersion matrix. n is the total number of samples. k is the number of clusters.

Together, they give a balanced view of how well the RobotStudio event sequences were grouped.

3

Methodology

This chapter starts by giving a description of the data-driven approach and providing data, then introduces the method to answer the first research question: Can meaningful user behavior patterns be identified from RobotStudio event logs? Then it is followed by the second research question: Can the discovered user behavior patterns be utilized to generate realistic test cases? The chapter is concluded by describing the Methodology used to answer the third research question: Can distinct and meaningful user behavior patterns be discovered through the clustering of RobotStudio event logs?

3.1 Data-driven approach

As briefly described in Section 1.1, the current approach to propose new RobotStudio features primarily relies on internal sources (e.g., own skilled employees, competitors and markets, testing of ideas), and external sources (e.g. from existing customers, customers of competitors, customers in adjacent areas to their offerings, suppliers, related industries) [9].

However, these methods lack a systematic framework for decision-making, which may lead to inefficiencies in feature development. Several challenges arise from this approach. One key limitation is that insights generated from these methods might be influenced by the personal biases or perspectives of the people involved in the process. Also, the global distribution of RobotStudio customers makes direct communication both time-consuming and logistically complex. Additionally, feature proposal decisions are not directly informed by actual user interaction data, increasing the risk of misalignment between development efforts and user needs. Moreover, the manual collection and processing of user demands place a significant strain on time and resources, ultimately reducing the efficiency of the feature development cycle.

To overcome these challenges, a more systematic and data-driven approach is required. Implementing a structured methodology for identifying and analyzing user interaction patterns will enhance the accuracy of demand detection and improve decision-making in feature prioritization. By leveraging user segmentation, pattern recognition analysis, and prioritization frameworks, the RobotStudio team can streamline feature development, optimize resource allocation, and ultimately improve overall user satisfaction.

3.2 Provided Data

To gain a comprehensive understanding of user interaction patterns, the event log files generated daily by RobotStudio are of great importance. A total of **1749** RobotStudio event log files (size of 63,943,195 bytes) were provided for this project are generated from RobotStudio version `24.2.10789.0` and sampled from July 2nd, 2024, exclusively from users with *Premium* license level. The RobotStudio event log files mainly record user interactions with the software. Each event log is stored in the `.rslog` file format, which consists of `HEADER` (header), `STATS` (system statistics), `SERVERDATA` (server data), and `LOG` (event sequence). The header provides essential information about the system, environment, and application state at the time of execution. The statistics section captures data on system resource usage during RobotStudio's operation, while the server data contains metadata related to the server or network environment. An example of event sequence is shown in Listing 3.1.

```
1  ----LOG BEGIN----
2  00:00:26.0199272|UIThreadWatchdog|Started
3  00:00:28.7417269|BackstageVisibilityChanged|False
4  00:00:29.0235611|UIThreadWatchdog|Warning|2000 ms delay
       on UI thread
5  00:00:29.0659201|ActiveWindowChanged|FileTextEditorWindow
       |NoId
6  00:00:29.1776178|RibbonTabChanged|RAPID
7  00:00:29.6732965|ActiveWindowChanged|DelayLoadedWindow|
       FileBrowser
8  00:00:29.6811704|ActiveWindowChanged|FileTextEditorWindow
       |NoId
9  00:00:33.5284104|ExecutingCommand|EditFind:
       FileTextEditorWindow|Shortcut
10 00:00:33.5986732|ExecutedCommand|EditFind:
       FileTextEditorWindow
11 00:00:38.3738859|ActiveWindowChanged|DelayLoadedWindow|
       SearchResults
12 00:00:40.5203074|ActiveWindowChanged|FileTextEditorWindow
       |NoId
13 00:00:40.5556198|ActiveWindowChanged|DelayLoadedWindow|
       SearchResults
14 00:00:45.3097260|ActiveWindowChanged|FileTextEditorWindow
       |NoId
15 00:00:48.5502295|ExecutingCommand|EditFind:
       FileTextEditorWindow|Shortcut
16 00:00:48.5962926|ExecutedCommand|EditFind:
       FileTextEditorWindow
17 00:04:33.7557792|ExecutingCommand|FileSave:
       FileTextEditorWindow|Activated
18 00:04:33.8202969|ExecutedCommand|FileSave:
       FileTextEditorWindow
```

```
00:04:37.7191437|UIThreadWatchdog|Warning|2000 ms delay
   on UI thread
----LOG END----
```

Listing 3.1: Representative Example Log Sequence

An event sequence can contain multiple lines of log entries. Typically, each log entry can include the following fields:

- **Timestamp:**
  Precise timing of each event, enabling sequence analysis.
  Example:
  This could be `00:00:26.0199272`, meaning the event occurred on 26.0199272 seconds after start up.
- **Event Type:**
  Descriptions of user actions, system states, or warnings.
  Examples:
  This could be `ActiveWindowChanged`, meaning there's a switch between windows in RobotStudio.
- **Detail:**
  - Contextual information about the event, such as tool usage, navigation between windows, configuration changes, command names, feature interactions, or error messages.
    Examples: This could be `FileTextEditorWindow|NoId`, together with `ActiveWindowChanged`, meaning that the user switches to File Text Editor Window.
  - Globally unique ID (GUID) for objects, controllers, or sessions.
    This could be *Example:* `08CA5D6D-CF9E-40ED-84A2-94346B7BBE5E`, which is a 128-bit label used to uniquely identify objects in computer systems [41].

Permission to access and utilize RobotStudio event log files has been granted. The dataset comprises logs from various versions of RobotStudio, multiple users, and different time periods. Therefore, the diverse and extensive log entries provide a rich foundation for understanding RobotStudio user interactions, identifying potential patterns, and possibly deriving actionable insights to improve the software functionality and user experience.

3.3 Data Preprocessing

Since user interaction is only related to `LOG` (event sequence), we decided to only focus on the event sequence of the RobotStudio event log file. As part of the data preprocessing pipeline, a custom function was developed to systematically clean and standardize the RobotStudio event sequence within the raw dataset. This section contains multi-line textual log entries generated during system events as shown in Listing 3.1. From this Listing 3.1, we can see that the most common format of

a single log entry is structured with `[Timestamp] | [Event Type] | [Detail] | ...`. The objective of the data preprocessing was to remove non-informative, redundant, or system-generated artefacts that could introduce noise into downstream analysis or modelling efforts. The following steps summarize the data preprocessing logic applied:

1. Removal of Exception Blocks and Error Trace Blocks
Entire segments of log messages encapsulated between predefined markers (`--EXCEPTION BEGIN-- and --EXCEPTION END--`) were removed. These often contain verbose debugging traces irrelevant to the structured event sequence analysis. However, the simple description of the exception before (`--EXCEPTION BEGIN-- and --EXCEPTION END--`) will be kept so the summary of exception can be known.

2. Line-by-Line Cleaning
Each message was processed line by line to perform the following sequence of regex-based removals and transformations:

- **Timestamps:** Leading timestamps of the format `1.00:00:13.2866918` or `00:00:26.0199272` were removed to ensure meaningful log sequence analysis.
- **Specific Patterns:** Patterns such as `-00:00:00.0070009 [10330]` or `00:00:00 [10357]`, which typically represent execution durations paired with process IDs, were stripped out.
- **Hexadecimal Codes:** System-level hexadecimal error or status codes (e.g., `0xc0048452`) were excluded to reduce low-level technical noise.
- **Globally Unique Identifiers (GUIDs):** Any 128-bit GUIDs (e.g., `566775ae-113d-4bea-a53b-6ad5d1f7a786`) were removed, as they do not contribute semantically to event categorization.
- **File Paths:** References to specific file paths such as `C:\path\to\file` were stripped, avoiding potential leakage of system details and ensuring privacy.
- **.sab File References:** File references ending in `.sab` were excluded.
- **Long Numeric Strings:** Numeric strings longer than 3 digits (likely to be IDs, large timestamps, or other non-semantic data) were removed.
- **Common File Types:** Each line was stripped of common file types with version info like `.dll (1.2.3.4)`.

3. Event Normalization
After the removal of blocks and line-by-line cleaning, some log entries still contain unnecessary details.

To further standardize the event data, we applied a rule-based normalization function, which transforms heterogeneous log messages into semantically meaningful and structurally consistent labels. This function uses predefined logic to retain only the most relevant arguments for each event type, thereby reducing noise and promoting comparability across events. For event types with no meaningful arguments (e.g., `NetworkAdapterChange`, `StationLoaded`), only the event name is

retained. For others, key arguments are appended to the event type using a delimiter (e.g., `ExecutedCommand|Copy`), enabling disambiguation between otherwise similarly named actions. Special cases such as `ActiveWindowChanged` and `VR` events are handled with additional logic to preserve context-sensitive information. For instance, `ActiveWindowChanged|DelayLoadedWindow|ControllerBrowser` will be normalized into `ActiveWindowChanged|ControllerBrowser`.

Each RobotStudio event log file's event sequences were then mapped according to this normalization function. After normalization, only the necessary details of each event type are kept, enhancing overall interpretability and convenience for future experiments.

4. Removal of Outlier Event Sequence Length To gain a deeper understanding of the nature of the event sequences in our log files, we analyzed the distribution of their lengths across 1,749 samples. The event sequence length is defined as the number of events (lines of log entries). Figure 3.1 shows a log-scaled distribution of event sequence lengths, with most sequences containing between 10 and 1,000 events, and a long tail of longer sequences.

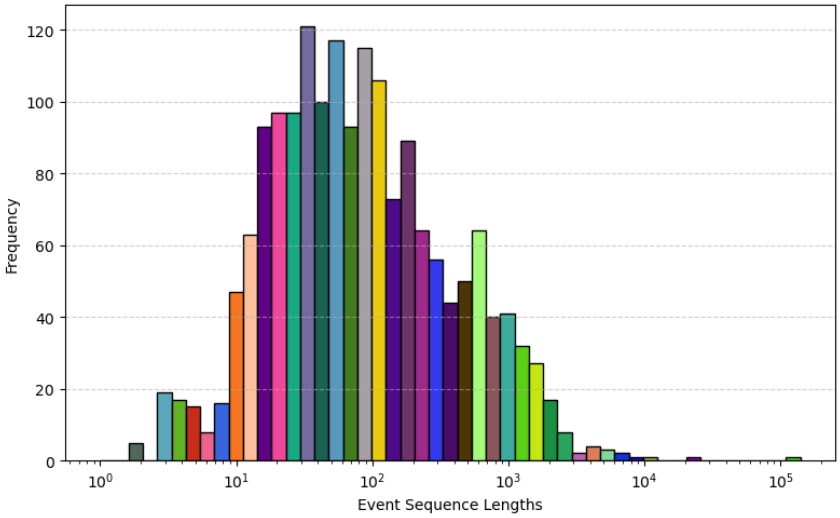

Figure 3.1: Log-scaled Distribution of Event Sequence Lengths

Table 3.1 shows key statistics of RobotStudio event sequence lengths. We can see that while the median sequence length is 73, the mean is substantially higher at approximately 361, driven upward by a small number of very long sequences. The maximum observed length reaches 140,038 events in a single file, which is more than two orders of magnitude beyond the third quartile (224), and nearly 400 times the median. This pronounced disparity between mean and median highlights the influence of long-tailed data and suggests that most logs are short, but some may record highly verbose sessions or edge-case behaviors.

To further clarify the distribution, we binned sequence lengths into coarse ranges in Table 3.2. Approximately 20.53% of the logs contain between 10 and 299 events, with the single largest bin being 30–99 events (31.7% of all logs). Only a small fraction—roughly 6.52%—have more than 1,000 events, and just one file exceeds

Table 3.1: Descriptive Statistics of RobotStudio Event Sequence Lengths

Metric	Value
Total Number of Event Sequences	1749.00
Minimum Event Sequence Length	2.00
25th Percentile (Q1)	27.00
Median Event Sequence Length (Q2)	73.00
75th Percentile (Q3)	224.00
Maximum Event Sequence Length	140038.00
Mean Event Sequence Length	361.59
Standard Deviation	3444.78

100,000. These observations confirm that, while most logs are relatively concise, a non-negligible tail of extreme cases exists, which may warrant special treatment or separate analysis.

Table 3.2: Proportion of Event Sequence Lengths

Event Sequence Length Range	Count	Proportion (%)
<10	102	5.83
10–29	375	21.44
30–99	555	31.73
100–299	359	20.53
300–999	244	13.95
1k–100k	113	6.46
>100k	1	0.06

We manually checked the RobotStudio event sequence length shorter than 10 lines and found that most of them are meaningless. Here in Listing 3.2 is one example of event sequence length < 10, and it basically recorded the user starting RobotStudio, having some add-ins automatically loaded and doing nothing:

```
1 ----LOG BEGIN----
2 00:00:00.7419851|AddinAdded|I/O Engineering|General
3 00:00:00.7419851|AddinAdded|Visual SafeMove 2|General
4 00:00:01.7525537|UIThreadWatchdog|Started
5 ----LOG END----
```

Listing 3.2: Example Log Sequence

Besides, the RobotStudio event sequence length longer than 100,000 lines has many duplicate log entries, which is not very useful for future experiments. Therefore, we can only consider middle-length RobotStudio event sequences, which range from 10 to 100,000 lines.

3.4 Pattern Mining

This section describes the methodology used to address RQ1 (Can meaningful user behavior patterns be identified from RobotStudio event logs?) and RQ2 (Can the discovered user behavior patterns be utilized to generate realistic test cases?). The approach integrates statistical modelling and graph structures, primarily through *n-gram* and *Markov chain* techniques, to identify user interaction patterns from RobotStudio log data. This method provides insights relevant to product design and test case generation.

Initially, user event sequences were segmented into fixed-length subsequences, known as *n-grams*, using a sliding window. A specialized analyzer tracked both the frequency of these patterns and their distribution across multiple user sessions. The identified *n-grams* were then ranked first by their consistency across sessions and secondarily by total frequency, highlighting significant behavioral patterns prevalent among diverse user interactions.

In addition, a first-order *Markov chain* model was employed to examine transition probabilities between events. Event transitions were recorded in a frequency matrix and normalized into conditional probabilities, capturing the likelihood of users transitioning from one event to another. These transition probabilities were represented visually through a weighted directed graph constructed using NetworkX, enabling intuitive visualization and supporting automated test case generation.

N-grams effectively detect frequent localized behaviors, while Markov chains clearly represent probabilistic event transitions, combining them together leverages their complementary strengths. This combined strategy efficiently captures complex user behaviors.

3.5 Log Context Clustering

This section describes the detailed methodology to answer RQ3: Can distinct and meaningful user behavior patterns be discovered through the clustering of RobotStudio event logs?

3.5.1 Log Context Embedding Generation

To cluster RobotStudio event sequences while preserving significant contextual and semantic information, we can generate embeddings based on the entire event sequence. A crucial step in this embedding generation is tokenization, the process of breaking down textual content into meaningful elements called tokens [42]. Consequently, we conducted a data analysis of RobotStudio event sequence tokens. For this, we applied a Bidirectional Autoregressive Transformer (BART) tokenizer to count these tokens in Table 3.3. The BART tokenizer is trained to split words effectively, treat spaces as integral parts of tokens, and separate punctuation and numbers into distinct tokens. For instance, the phrase “Hello world!” is tokenized into five elements: “”, “Hello”, “world”, “!”, and “”. However, BART can only handle up to 1024 tokens, which

is lower than the maximum number of tokens in the RobotStudio event sequence. In Figure 3.2 about event sequence length statistics, we can conclude that 90% RobotStudio event sequences are below 715 lines. Likewise, in Table 3.3, the majority of event sequence lengths lie between 10 lines and 1000 lines.

Table 3.3: Summary of RobotStudio Event Sequence Length Statistics

Metric	Value
Total Number of RobotStudio Event Sequences	1749.00
Min Event Sequence Length	2.00
Max Event Sequence Length	140,038.00
Mean Event Sequence Length	361.59
Median Event Sequence Length	73.00
90th Percentile of Event Sequence Length	714.20

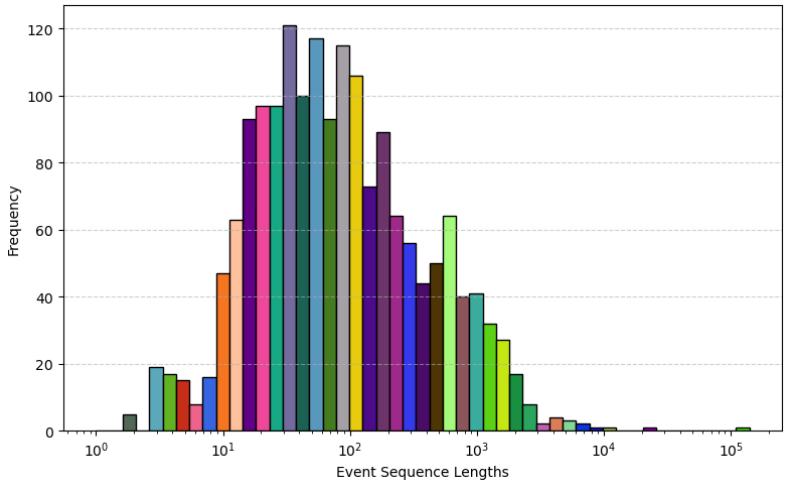

Figure 3.2: Log-scaled Distribution of RobotStudio Event Sequence Length

To address the challenge of long sequences in RobotStudio event log files, we adopted the Longformer Encoder-Decoder (LED) architecture, as introduced in Subsection 2.4.3, for generating embedding representations. While traditional models like BERT (512 tokens) and standard BART (1024 tokens) would be insufficient and computationally expensive for this task, `LED` supports significantly longer sequences, up to 16,384 tokens, making it suitable for capturing the extensive context within log entries. Notably, the LED tokenizer is an alias of the BART tokenizer, ensuring a consistent token distribution. The subsequent steps describe our method for generating log context embeddings for RobotStudio event sequences.

3.5.1.1 Tokenization and Chunking

Due to the token length limit of the LED model, a sliding window strategy was employed for processing input sequences. First, each preprocessed event sequence was tokenized using `LEDTokenizerFast` without truncation [32]. Then, text exceeding the max sequence length of 16,384 tokens was processed in overlapping chunks using

a stride of 4,096. It means each part of the chunk was 16,384 tokens long, and we moved forward by 4,096 tokens each time, like sliding a window over the text with a small step. Here in Table 3.4, we can conclude that 90% of RobotStudio event logs are below 5450 tokens. The `LED` model was employed to reduce the truncation for log entries as much as possible, especially for most event log files under 16,384 tokens, shown in Figure 3.3.

Table 3.4: Summary of RobotStudio Event Sequence Token Count Statistics

Metric	Value
Total Number of RobotStudio Event Sequence	1749.00
Min Token Count	17.00
Max Token Count	840,309.00
Mean Token Count	2613.16
Median Token Count	536.00
90th Percentile of Token Count	5449.60

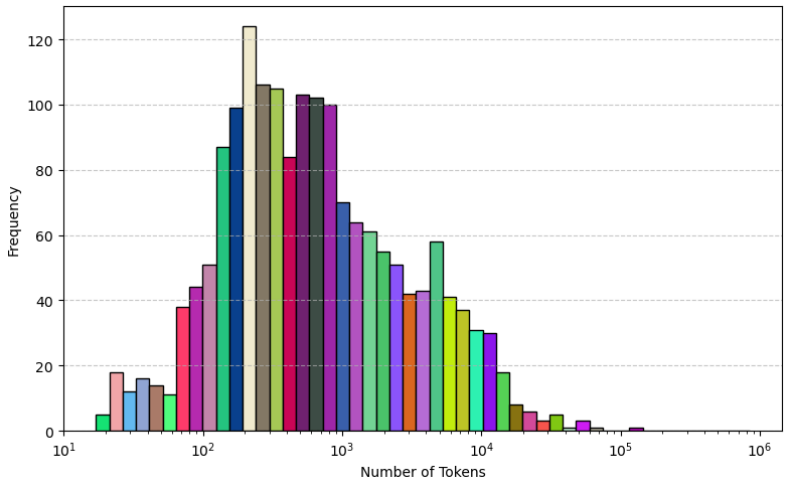

Figure 3.3: Distribution of RobotStudio Event Sequence Token Count

3.5.1.2 Attention Masking

Self-attention, sometimes called intra-attention, is an attention mechanism that relates different positions within a single sequence to compute a representation of that sequence [43]. In essence, it tells the model which parts of the input sequence it should treat as "important" and which parts it should effectively ignore during the computation of attention scores. Following tokenization and chunking, to ensure consistent input lengths for the model, we padded all chunks to a fixed size of 16,384 tokens using a dedicated "empty" token. To enable the model to distinguish between actual content effectively and this added padding, we employed an attention masking mechanism. Furthermore, to facilitate the processing of long text, the `LED` model was configured with global attention on the first token of each input chunk, allowing this initial token to attend to all other tokens in the input. This strategy significantly enhances the `LED` model's ability to understand extended textual sequences.

3.5.1.3 Embedding Extraction

For each chunk of log entries, we used a pre-trained `LED` model to transform the tokens into vectors. Then, we calculated the average of all the token vectors in that chunk to obtain a single vector that represents the entire chunk.

If a RobotStudio event sequence is too long and split into several chunks, we average all those chunk vectors to get one final vector. This final embedding vector serves as a numeric summary of the entire log message, capturing both the important context from small details and the broader picture.

3.5.2 Clustering Based On Embeddings

The following steps show how we clustered RobotStudio event sequences based on their context embeddings:

To achieve better clustering performance, the elbow method was applied to the log context embeddings generated from preprocessed event sequences in RobotStudio. The elbow method is a heuristic approach used to determine the optimal number of clusters in a dataset, first introduced in 1953 [44]. In clustering analysis, data points are grouped into clusters based on their similarity to one another. A standard metric used to evaluate the quality of these clusters is the explained variance, also referred to as the distortion score. This metric measures the average distance of data points within a cluster from the cluster's centroid, indicating how compact or tightly grouped the points are.

Initially, increasing the number of clusters significantly reduces the within-cluster distances, as data points become more tightly grouped around their respective centroids. Consequently, the explained variance improves substantially. However, beyond a certain point, adding additional clusters yields diminishing returns, and the rate of improvement slows considerably. This inflection point, which resembles an"elbow" on a graph of explained variance versus the number of clusters, is commonly used to estimate the optimal number of clusters. This approach is known as the elbow method.

To gain a comprehensive understanding of RobotStudio event log files and to minimize the need for extensive manual inspection, the elbow method was applied using K-Means clustering. In our experiments, we explored clusters ranging from 2 to 100 to identify a suitable clustering configuration.

After determining the optimal number of clusters using the elbow method, we proceeded by experimenting with various clustering algorithms. Because the log context embeddings generated are high-dimensional, the following clustering algorithms are selected: K-Means clustering, agglomerative clustering, Gaussian Mixture Model (GMM) clustering, and spectral clustering, as mentioned in Subsection 2.4.4. After clustering, the same N-gram procedure will be applied to each cluster of event sequences to see the distinct user behavior patterns across different groups.

4

Results

This chapter begins with statistical analysis, followed by global pattern mining results and then N-gram results per cluster.

4.1 Statistical Analysis

In order to have a better understanding of the RobotStudio event sequence before experiments, we did a statistical analysis of the event sequence after data preprocessing.

4.1.1 Event Type Frequency

We analyzed the frequency of each event type (without details) in RobotStudio log files (`.rslog`). A total of 37 unique event types were identified. A word cloud is a visual representation of text data where the size of each word indicates its frequency. In Figure 4.1, the size of the text represents the frequency of event type. From Figure 4.1, we can observe that `ActiveWindowChanged`, `UserLogMessage`, `ExecutingCommand`, and `ExecutedCommand` preoccupied the word cloud, indicating high frequency across all event types.

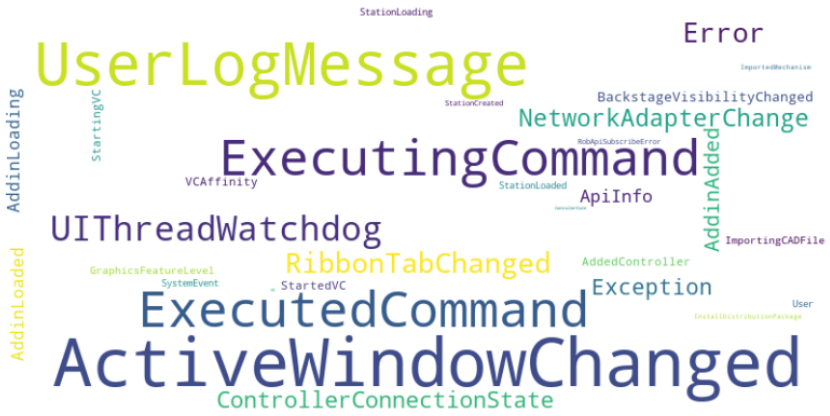

Figure 4.1: Word Cloud of Event Types

Figure 4.2 lists the top 30 most frequent event types. From Figure 4.2, we can see that `ActiveWindowChanged`, `UserLogMessage`, `ExecutingCommand`, and `ExecutedCommand`

each occurred more than 60,000 times. These top events reflect frequent user interactions with key UI components or background system checks. Most event types occur only a few times, while a small number of them appear very frequently. Statistics show that the median frequency is 1851, while the maximum occurrence of event type `ActiveWindowChanged` is 115,538. This skewed distribution suggests that the event sequence is dominated by a few repeated event types.

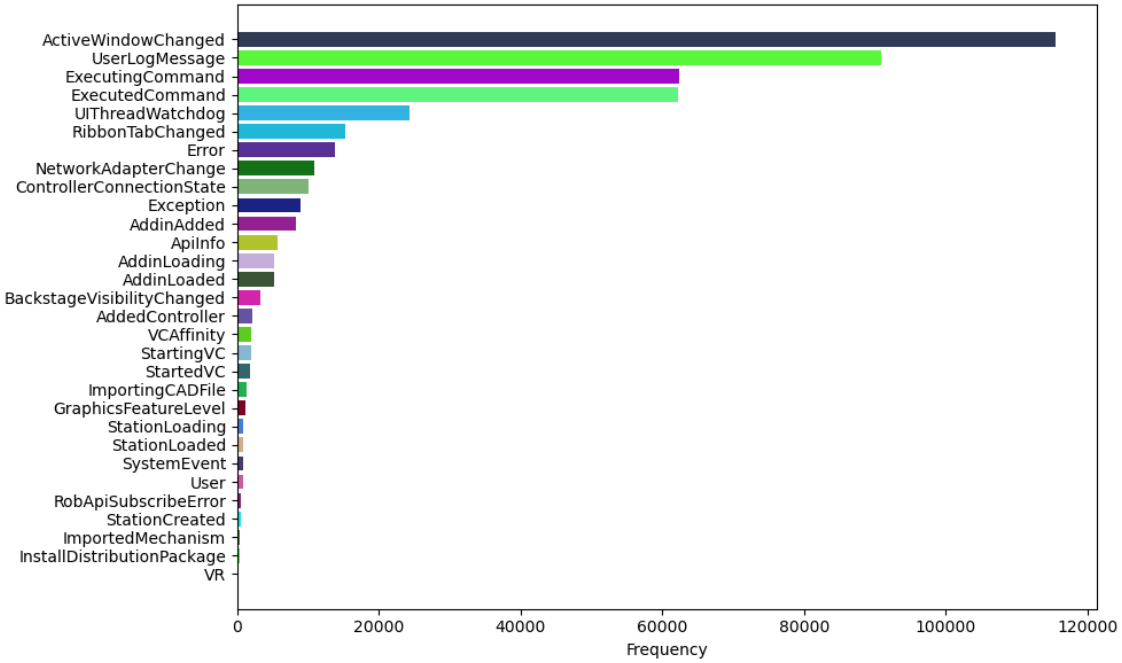

Figure 4.2: Top 30 Most Frequent Event Types

4.1.2 Distributions of System Event and User Event's Occurrence

To better understand the behavioral composition of the logs, we conducted a comparative analysis of event frequency distributions for system-level and user action events. User actions were identified via a prefix-based filter, using the following identifiers: `ExecutingCommand`,`ExecutedCommand`, `ImportedMechanism`, `ExportGeometry`, `VR`, `ActiveWindowChanged`, and `RibbonTabChanged`. All other event log entries were treated as system-level.

Figure 4.3 shows the distribution of system event occurrences. It exhibits an extremely long-tail pattern with a small number of high-frequency system events dominating, while the majority appear only a handful of times.

A similar structure appears in Figure 4.4, though the overall frequency levels are higher. Notably, system events include outliers reaching 187,642 occurrences, whereas user actions peak at 25,376. Additionally, user events are more evenly distributed in the low-frequency ranges, possibly reflecting a broader diversity of interaction types.

These differences suggest that while both event categories share long-tailed character-

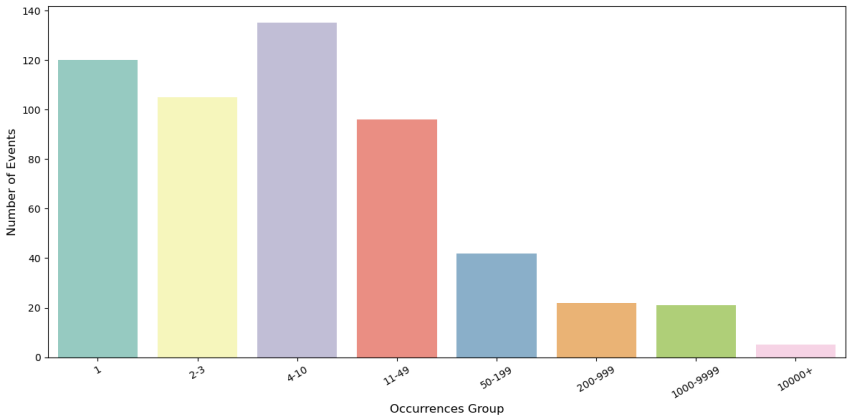

Figure 4.3: Distribution of System Event Occurrences

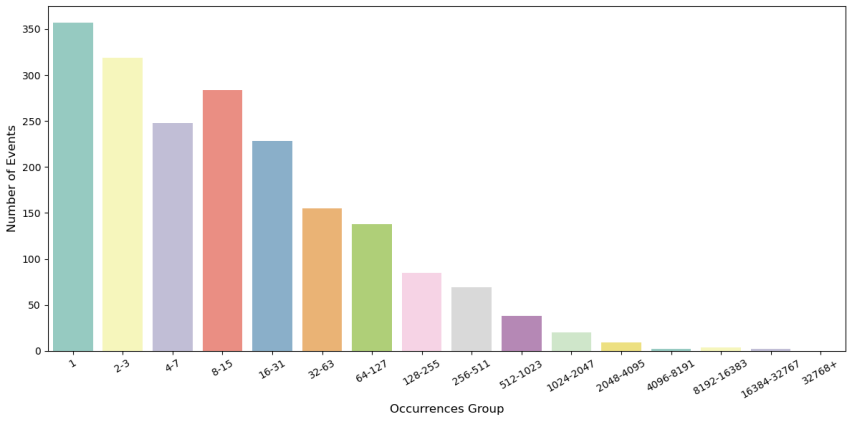

Figure 4.4: Distribution of User Event Occurrences

istics, system events are sparser and log-intensive, whereas user actions are repetitive and semantically richer. This highlights the need for differentiated treatment in downstream tasks such as modelling and feature extraction.

4.2 Global Pattern Mining Results

To investigate common user interaction patterns, we extracted sequential event patterns, also known as N—gram, from an original dataset of 1,749 RobotStudio log files (`.rslog`). To ensure the representativeness and manageability of the dataset, we further filtered out sequences whose length was not within the range of 10 to 3000 events, resulting in a refined dataset of 1644 logs suitable for detailed analysis. The objective of this analysis is to uncover frequently occurring user action sequences, thereby identifying typical workflows and usage habits.

An N-gram is defined as a consecutive sequence of N events occurring in a fixed order. For example, a 4-gram might represent the following interaction flow: `Open Project` → `Edit Code` → `Run Simulation` → `Save Project`.

We analyzed two distinct subsets of N-gram data: **Filtered Sequences** and **Unfiltered Sequences**. The `Filtered Sequences` subset consists only of events beginning with specific user action-related prefixes (`"ExecutedCommand", "ImportedMechanism", "ExportGeometry", "VR", "ActiveWindowChanged", "RibbonTabChanged"`). Conversely, the `Unfiltered Sequences` subset includes all recorded events without restrictions.

Each N-gram was evaluated based on two key metrics:

- **File Coverage**: The count of unique log files containing the N-gram, indicating how widely a pattern is shared among different users.
- **Total Occurrences**: The cumulative frequency of the N-gram across all logs, including multiple occurrences within a single session.

The sequences in our analysis are primarily ranked by their **Total Occurrences**, emphasizing patterns that reflect common and repetitive workflows. This filtering approach effectively isolates meaningful user-triggered actions from background system events, thus sharpening the focus on behaviorally relevant interaction sequences.

4.2.1 Abbreviation Key

After clustering the event sequence, and to ensure clarity and comply with confidentiality policies, event type labels in the results were abbreviated. The mapping of the original event type to abbreviated event type is shown in Table A.1.

4.2.2 Filtered N-gram Analysis

To identify common user interaction patterns, we analyzed 4-gram, 6-gram, and 8-gram sequences derived from a filtered dataset containing 1644 RobotStudio log

files. We focused specifically on high-frequency sequences that appeared across multiple sessions to capture representative user workflows and navigation behaviors.

The most frequent interaction pattern involves continuous usage of the `RapidEditor Window`. For example, sequences where the user repeatedly switches within this window appeared prominently across all sequence lengths, including 4-gram (present in 196 files, 4090 total occurrences), 6-gram (115 files, 3141 total occurrences), and 8-gram (76 files, 2718 total occurrences). This suggests extensive and intensive user engagement in code editing tasks without frequent interruptions.

Another prevalent behavior identified was the frequent alternation between `RapidEditor Window` and `ControllerBrowser`. For instance, the 4-gram sequence (`RapidEditor Window → ControllerBrowser → RapidEditorWindow → ControllerBrowser`) appeared in 74 distinct log files, totaling 2174 occurrences. Such repetitive toggling indicates users regularly switch between code editing and controller management, possibly to verify controller states or integrate device logic into their code.
Sequences involving the `ExecutedCommand|RapidEditorShow` command also occurred frequently. A representative example (`ControllerBrowser → RapidEditor Window → RapidEditorShow → ControllerBrowser`) was observed in 315 files with 1883 occurrences. Users likely use this command not only to initially open the editor window but also to re-establish focus or restore the workspace view after context switches.

Additionally, consistent alternations between the `GraphicWindow` and the `Element Browser` were observed. For example, the 4-gram pattern (`GraphicWindow → Element Browser → GraphicWindow → ElementBrowser`) appeared in 106 files, totalling 1642 occurrences. This repeated navigation suggests iterative tasks involving graphical element selection, inspection, or debugging.
Overall, the analysis uncovers several notable patterns of user interaction. Users predominantly spend their time within specific windows, particularly the code editing environment, which implies a highly concentrated and task-driven workflow. Nevertheless, there is substantial evidence of frequent window switching, especially between editor views and various controller-related interfaces. Such frequent transitions might indicate potential usability challenges or reveal insufficient integration between different UI components.

Additionally, the regular use of commands like `RapidEditorShow` to regain or manage focus suggests that users often encounter difficulties with window-handling mechanisms. This frequent manual intervention highlights the necessity for more intuitive or automated window management solutions.

Furthermore, recurring shifts between graphical views and element browsers underscore the regular engagement in tasks related to graphical configuration and visual debugging. Recognizing this usage pattern emphasizes the importance of refining UI components dedicated specifically to these frequent tasks.

To sum up, these identified interaction patterns provide crucial guidance for enhancing RobotStudio's user interface design. Insights from real user behaviors can

facilitate targeted improvements, increase overall workflow efficiency, and inform the development of realistic and effective test cases.

Table 4.1: Top 10 4-gram by Total Occurrences (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW ×4	196	4090
2	(AWC|REW → AWC|CB) ×2	74	2174
3	(AWC|CB → AWC|REW) ×2	81	2079
4	AWC|CB → AWC|REW → EC|RES → AWC|CB	315	1883
5	AWC|REW → EC|RES → AWC|CB → AWC|REW	297	1823
6	EC|RES → AWC|CB → AWC|REW → EC|RES	288	1758
7	(AWC|GW → AWC|EB) ×2	106	1642
8	(AWC|EB → AWC|GW) ×2	88	1349
9	(AWC|GW → AWC|OB) ×2	145	895
10	(AWC|TCW → AWC|GW) ×2	130	826

4.2.3 Unfiltered N-gram Analysis

Analyzing the unfiltered event sequences from 1644 log files reveals that the most frequently occurring patterns primarily reflect system-level activities rather than direct user interactions. Highly repetitive sequences include:

- `UserLogMessage|Warning → UserLogMessage|Warning → ...`, appearing in 508 files with 35,975 occurrences.
- Continuous sequences of `Error` events, noted in 74 files, totalling 11,077 occurrences.
- `UIThreadWatchdog|Warning → UIThreadWatchdog|Warning → ...`, found in 282 files with 11,064 occurrences.
- Frequent `UserLogMessage|Error` and `NetworkAdapterChange` events, each accumulating several thousand occurrences.

Such repetitive patterns typically reflect automated system activities, including logging, system monitoring, network status updates, and recurring error reporting. Although crucial for diagnosing system health, they offer minimal insight into user-driven behavior.

Despite the dominance of these system-level events, meaningful user interaction sequences still emerge clearly. Notable interactive patterns involve deliberate navigation and command executions, such as:

- `ActiveWindowChanged|ControllerBrowser` → `ExecutingCommand|RapidEditorShow` → `ActiveWindowChanged|RapidEditorWindow` → `ExecutedCommand|RapidEditorShow`
  This sequence is observed in 508 files (35,975 occurrences). This sequence suggests consistent user workflows involving controller interactions and code editing.

- Repeated switching within the `RapidEditorWindow`, noted in 188 files with 3,301 occurrences, indicating prolonged editing tasks without interruption.

These user-driven patterns differ significantly from system-generated logs. They are typically shorter, task-oriented, and indicative of deliberate interface interactions rather than background operations. Recognizing such patterns helps identify opportunities to streamline workflows, optimize UI design, and prioritize targeted system improvements.

Table 4.2: Top 10 4-gram by Total Occurrences (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn ×4	508	35975
2	Err ×4	74	11077
3	UITW|Warn ×4	282	11064
4	ULM|Err ×4	265	8359
5	NAC ×4	257	5428
6	AWC|CB → EXC|RES → AWC|REW → EC|RES	505	4696
7	EXC|NRE ×4	14	3848
8	AWC|REW ×4	188	3301
9	API ×4	62	2867
10	EXC|RES → AWC|REW → EC|RES → AWC|CB	315	1902

4.2.4 Summary

Comparing the unfiltered and filtered N-gram results reveals distinct differences in user interaction patterns. Initially, before applying any filtering, the most common N-gram predominantly consisted of automated system events, such as repeated warnings (`UserLogMessage|Warning`), error notifications, and watchdog alerts (`UIThreadWatchdog|Warning`). After filtering, these system-generated events are excluded, allowing a clearer view of intentional user actions, such as deliberate window switching and specific command executions.

Moreover, the filtered results offer a more explicit representation of task-oriented behaviors. A noticeable pattern emerges, showing repetitive switching between particular windows, especially between `RapidEditorWindow` and `ControllerBrowser`. This frequent alternation implies that users consistently transition between editing tasks and monitoring or managing controller settings, reflecting structured workflows rather than random or arbitrary interactions.

Additionally, structured interaction flows become more evident in the filtered sequences. There are clear, repeating patterns in user behaviors, notably the regular use of the command `RapidEditorShow` following window changes. Such patterns suggest deliberate user efforts to maintain or regain interface focus and to restore workspace configurations, possibly due to existing interface design or usability challenges.

In conclusion, filtering out background noise significantly enhances the visibility of authentic user behaviors. This clarity provides practical insights, aiding efforts to improve the user interface design and streamline overall workflow efficiency.

4.3 Markov Chain Event Transition

To capture realistic user behavior in system testing, we constructed a first-order Markov chain based on event transition logs extracted from RobotStudio. Each unique system event was modelled as a state, with transition probabilities computed from empirical frequencies. The resulting transition matrix includes a total of 1139 states and 5,294 non-zero transitions, with each state leading on average to 4.65 distinct successors.

4.3.1 Global Structure of the Markov Chain Model

To gain a comprehensive understanding of the modelled system behavior, we analyzed the overall structure of the constructed first-order Markov chain. This includes statistics on the number of states, transitions, and the average branching factor per state.

The transition matrix consists of **1,139** distinct states, corresponding to unique system events observed in user interaction logs. Between these states, a total of **5,294** transitions were recorded, each representing a non-zero probability of moving from one event to another.

Table 4.3: Markov Chain Summary: Key Statistics

Metric	**Value**
Total unique events (nodes)	1139
Total transitions (edges)	5294
Average out-degree per node	4.65

Comparing the unfiltered and filtered N-gram results reveals distinct differences in user interaction patterns. Initially, before applying any filtering, the most common N-gram predominantly consisted of automated system events, such as repeated warnings (`UserLogMessage|Warning`), error notifications, and watchdog alerts (`UIThreadWatchdog|Warning`). After filtering, these system-generated events are excluded, allowing a clearer view of intentional user actions, such as deliberate window switching and specific command executions.

Moreover, the filtered results offer a more explicit representation of task-oriented behaviors. A noticeable pattern emerges, showing repetitive switching between particular windows, especially between `RapidEditorWindow` and `ControllerBrowser`. This frequent alternation implies that users consistently transition between editing tasks and monitoring or managing controller settings, reflecting structured workflows rather than random or arbitrary interactions.

Additionally, structured interaction flows become more evident in the filtered sequences. There are clear, repeating patterns in user behaviors, notably the regular use of the command `RapidEditorShow` following window changes. Such patterns suggest deliberate user efforts to maintain or regain interface focus and to restore workspace configurations, possibly due to existing interface design or usability challenges.

On average, each state connects to approximately **4.65** successor states, suggesting that user and system behaviors exhibit moderate complexity and context dependency. This branching factor indicates a rich set of user interaction possibilities and varying system responses, highlighting the diverse pathways available within the system.

The branching structure implies several important insights. First, the system enables multiple workflows depending on user goals and context. Second, many interface states function as transition hubs, providing numerous possible outcomes. Third, the complexity of the event space demands structured sampling approaches for effective exploration. Lastly, probabilistic traversal methods, such as weighted random walks, are appropriate for simulating user interactions and generating realistic test cases.

Given the substantial number of transitions (5294 across 1139 nodes), exhaustive path testing becomes impractical. This limitation underscores the need for prioritization strategies, where transitions are selectively tested based on frequency, criticality, or risk, aligning testing efforts closely with real-world usage patterns.

4.3.2 High In-Degree States and Behavioral Convergence

We also investigated the most frequently targeted event states—those with the highest number of distinct incoming transitions. These nodes act as convergence points in user or system behavior, indicating views or states that are revisited from many different paths.

Table 4.4 lists the top 10 nodes with the highest in-degree. At the top is `ActiveWindowChanged|GraphicWindow`, with **506** incoming edges, confirming its role as a central interface for visual interactions or simulation. This makes it a key candidate for GUI regression testing, especially under diverse entry conditions.

Other commonly revisited states include:

- `ActiveWindowChanged|ObjectBrowser`, `ElementBrowser`, and `ToolControlWindow`, indicating frequent inspection or configuration actions;
- `RibbonTabChanged|RAPID`, `Controller`, and `Home`, reflecting users' frequent switching across UI modules;
- `ActiveWindowChanged|Output` and `RapidEditorWindow`, suggesting common destinations following task execution or code edits.

From a testing perspective, these states deserve special attention. Since they are reached from many different paths, it's essential to verify that the system behaves consistently regardless of the user's approach. Frequent navigation to these views also makes them prime spots for UI glitches or performance slowdowns, so tests should ensure everything loads correctly and runs smoothly even after repeated use. These areas are also more likely to reveal problems caused by leftover state from earlier actions.

Table 4.4: Top 10 High In-Degree Nodes in Event Transition Graph

Event	**In-Degree**
ActiveWindowChanged|GraphicWindow	506
ActiveWindowChanged|ObjectBrowser	276
ActiveWindowChanged|ElementBrowser	256
ActiveWindowChanged|Output	244
ActiveWindowChanged|RapidEditorWindow	235
ActiveWindowChanged|ControllerBrowser	227
RibbonTabChanged|RAPID	173
ActiveWindowChanged|ToolControlWindow	165
RibbonTabChanged|Controller	162
RibbonTabChanged|Home	124

4.3.3 Deterministic Transitions ($P = 1.0$)

Some transitions in the event graph occurred with a probability of exactly 1.0, meaning the target event always followed the source event whenever it occurred. These deterministic transitions highlight fixed behavioral chains within the system. Certain commands consistently lead to specific view transitions, such as opening a tool window or navigating to a configuration interface. Additionally, events associated with importing robots, tools, or mechanisms systematically trigger predictable backend operations, such as loading libraries or updating models. Commands related to editing or resolving references, like `RapidEditorFindUnusedRefs`, typically result in an automatic return to the editor window, forming closed interaction loops. Furthermore, system-driven actions such as path editing, simulation, or collision analysis often involve tight integration between command execution and subsequent interface changes, creating deterministic pathways.

From a testing perspective, transitions with a probability of 1.0 are particularly useful. They can be used to identify potential breakpoints or errors in highly predictable flows, or serve as high-confidence paths in behavioral modelling and support the generation of test cases that target critical transitions, particularly in UI logic or view-switching workflows.

4.3.4 Summary

By sampling paths from the Markov model, we can generate representative test cases that reflect natural usage frequencies. High-probability paths focus on covering dominant behaviors, while rare transitions are useful for robustness testing.

Markov chains can serve not only as a behavioral modelling tool in test case generation, but also as a semantic foundation for reinforcement learning (RL) and intelligent fuzzing strategies. By capturing the probabilistic structure of user interactions, they provide valuable prior knowledge and help constrain exploration in a meaningful way.

In reinforcement learning, the underlying assumption is often modelled as a Markov

Decision Process (MDP), where the next state depends solely on the current state and action. When explicit environmental feedback is limited or testing goals are not well-defined, a pre-constructed Markov chain can guide the agent's initial policy or reward structure. Transition probabilities derived from user logs can be used to bias the agent's action preferences toward behavior patterns observed in practice. During the early stages of training, this kind of prior-driven exploration can help avoid ineffective or unrealistic action sequences, ultimately improving convergence and test relevance.

In the context of intelligent fuzzing, Markov chains offer a principled approach to guiding input generation based on observed execution patterns. While traditional fuzzers often rely on random mutations, a Markov-guided fuzzer can generate event sequences with a higher likelihood of exercising critical system behaviors. By leveraging historical transition probabilities, the fuzzer can prioritize sequences that are more likely to cause state changes, increase code coverage, or trigger subtle faults. This approach is especially useful in GUI-based testing or command sequence validation, where user-like navigation paths can be reproduced to maximize test effectiveness.

Taken together, the use of Markov chains offers a flexible bridge between data-driven and model-based test generation. They enhance interpretability, reinforce real-world relevance, and strike a better balance between exploration and precision in automated testing strategies.

4.4 Log Embedding Clustering

The embedding task was performed on a high-performance computing environment comprising 6 vCPUs (Intel Xeon E5-2690 v4), 112 GB RAM. The system was equipped with one NVIDIA Tesla V100 GPU (16 GB) for acceleration. The clustering task was performed on a stationary computer equipped with 16 CPUs (i9-12900F), 32 GB of RAM, and an NVIDIA GeForce RTX 3070.

4.4.1 Optimal Number of Clusters

From Figure 4.5, we can see that the turning point of the explained variance (distortion score) from 2 to 100 is **11**.

4.4.2 Evaluation of Clustering Algorithms

After determining the optimal number of clusters, we can proceed with evaluating different clustering methods based on log context embedding. From Table 4.5, we can see that it compares clustering algorithms based on performance metrics. Spectral clustering achieved the highest Silhouette Score and lowest Davies-Bouldin Index, indicating superior clustering quality. K-Means yielded the highest Calinski-Harabasz Score, suggesting good cluster separation. GMM performed moderately. Agglomerative clustering showed balanced performance across metrics.

After examining the performance of the above clustering algorithms, to have a

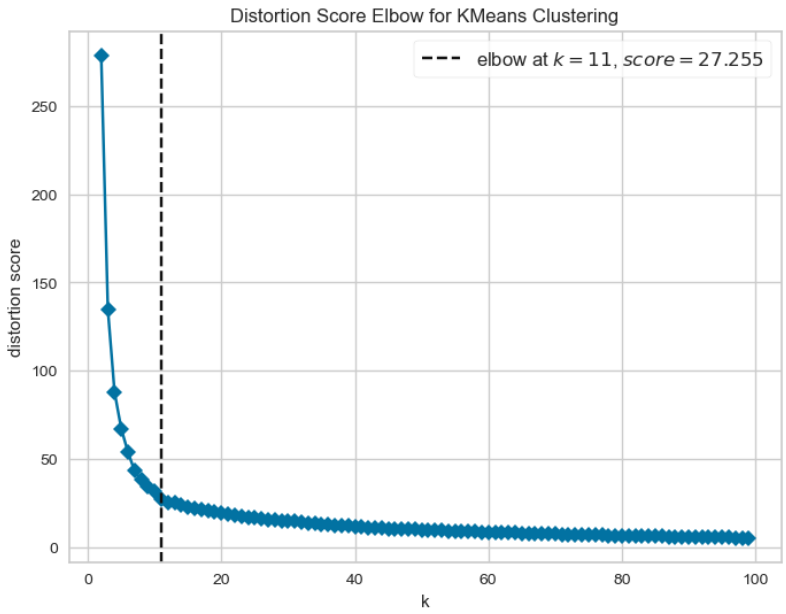

Figure 4.5: Optimal Number of Clusters Using Elbow Method

Table 4.5: Clustering Performance of Different Clustering Algorithms

#	Algorithm	Silhouette Score	Davies-Bouldin Index	Calinski-Harabasz Score
1	K-Means	0.510484	0.869354	4585.07
2	Agglomerative	0.466536	0.909279	4173.49
3	GMM	0.510744	0.869174	4583.50
4	Spectral	0.721476	1.020210	196.57

clearer picture of the clustering results in 2-dimensional space, we applied t-SNE visualization to clustered high-dimensional embeddings for each clustering algorithm. T-SNE is an iterative algorithm for visualizing high-dimensional data by mapping the data points to a two- or finite-dimensional space [45].

All t-SNE visualization points form a long, curved "U" or spiral-like shape, which is typical of non-linear separable data. In Figure 4.6, Figure A.1, and Figure A.2, the distributions of clusters in t-SNE visualization are more even than in Figure 4.7. For Figure 4.6, Figure A.2, and Figure 4.7, the majority of the distribution of clustering results are in cluster 0. For Figure A.1, the majority of the distribution of clustering results are in cluster 5. In Figure 4.7, the majority of the distribution of clustering results is in cluster 0. This uneven distribution suggests that spectral clustering captures the global structure differently. This is likely due to its graph-based approach, which is more sensitive to data topology than centroid- or density-based methods.

Combined with the clustering performance and t-SNE 2-dimensional visualization of each algorithm's clusters, we can conclude that K-Means is the best algorithm in the Calinski-Harabasz score and has the best visualization. In Figure 4.6, clusters 5, 9, 0, 6 are well-separated, but others are less clearly separated, often gradually transitioning from one to another.

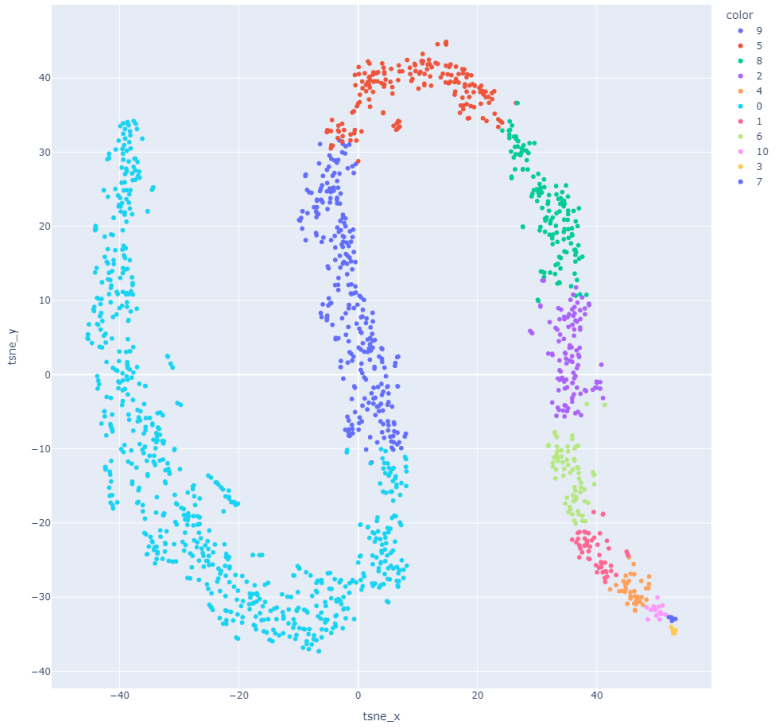

Figure 4.6: T-SNE Visualization of K-Means Clustering Result

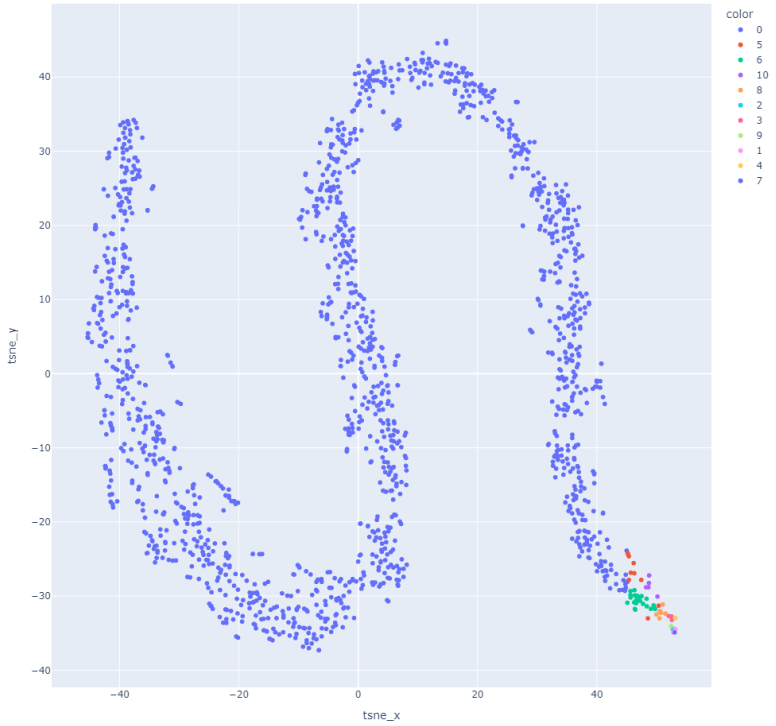

Figure 4.7: T-SNE Visualization of Spectral Clustering Result

4.4.3 N-gram Analysis Per Cluster

4.4.3.1 Filtered N-gram Analysis Per Cluster

To get a better understanding of each cluster's distinct user behavior patterns, we examined the top 10 N-gram sequences for major clusters, including cluster 0, 2, 5, 8, and 9. Because these clusters have the top 5 file counts in total distribution of files, far higher than other clusters, as in Table 4.6.

From Table A.6, Table A.7 and Table A.8 in cluster 0, the dominant 4-gram, 6-gram and 8-gram sequences mainly involving switching to the Controller Browser (`ActiveWindowChanged | ControllerBrowser`), then focusing on the Rapid Editor Window (`ActiveWindowChanged |RapidEditorWindow`), followed by executing a command to show/refresh the Rapid Editor (`ExecutedCommand | RapidEditorShow`), and potentially returning to controller management. This suggests a common workflow of navigating to the controller management, opening or refreshing the code editor, and potentially returning to the controller management. To improve this workflow, RobotStudio could offer enhanced navigation between these core areas, perhaps through direct links or more integrated views. We can also see recurring sequences involving switching to the Graphic Window (`ActiveWindowChanged | GraphicWindow`) and then the Object Browser (`ActiveWindowChanged | ObjectBrowser`). This indicates another frequent interaction between visualizing the simulation and inspecting project objects. When users transition between simulation and objects, providing relevant information in the new

Table 4.6: Distribution of Files Across Clusters Using K-Means Clustering

Cluster	File Count
0	747
1	51
2	112
3	6
4	38
5	178
6	72
7	6
8	137
9	282
10	15

context could enhance user satisfaction.

From Table A.12, Table A.13, Table A.14, cluster 2 is dominated by alternating sequences of `ActiveWindowChanged | GraphicWindow` → `ActiveWindowChanged | ElementBrowser` and `ActiveWindowChanged | RapidEditorWindow` → `ActiveWindowChanged | ControllerBrowser`. These suggest two distinct but highly repetitive interactions within the files. RobotStudio could benefit from offering optimized layouts with simultaneous use of the Graphic Window and Element Browser, as well as the Rapid Editor with the `Controller Browser`. Also, the presence of sequences involving `ExecutedCommand | RapidEditorShow` alternated with `ActiveWindowChanged | ControllerBrowser` and `ActiveWindowChanged | RapidEditorWindow` is also significant, indicating a flow or dependency between these elements. The same applies to cluster 0; RobotStudio could enhance navigation between them. The highly repetitive `ActiveWindowChanged | SisAnalyzer.SisFileList` → `ActiveWindowChanged | SisAnalyzer.SummaryTable` sequences, despite very low file coverage, are notable for their intensity within the single file in which they appear. Given this intense, albeit isolated, usage, exploring ways to make the SisAnalyzer workflow more integrated or accessible could be valuable.

From Table A.21, Table A.22 and Table A.23, cluster 5 is dominated by the prominent `ActiveWindowChanged | ControllerBrowser` → `ActiveWindowChanged | RapidEditorWindow` → `ExecutedCommand | RapidEditorShow` sequence. This suggests a key code editing workflow is central to this cluster. To better support this, RobotStudio could further enhance the Rapid Editor with improved code completion, error highlighting, and debugging tools. Also, alternating sequences of switching between Graphic Window (`ActiveWindowChanged | GraphicWindow`) and Object Browser (`ActiveWindowChanged | ObjectBrowser`), and switching to the Tool Control Window (`ActiveWindowChanged | ToolControlWindow`) and then the Graphic Window also appear, indicating other less dominant but present interactions. Lastly, alternating Rapid Editor Window and Controller Browser sequences are present but less dominant than in Clusters 3 and 4. If tool management is a significant secondary

task, ensuring the Tool Control Window is easily accessible and integrated with the 3D environment would be beneficial.

From Table A.30, Table A.31 and Table A.32, cluster 8 is dominated by the `ActiveWindowChanged | ControllerBrowser` → `ActiveWindowChanged | RapidEditorWindow` → `ExecutedCommand | RapidEditorShow` sequence. The high frequency and file coverage of this sequence indicate that code editing and controller management are the primary activities in this cluster. Similar to cluster 6, there is a significant amount of switching to the Search Results window (`ActiveWindowChanged | SearchResults`). Use direct navigation between the `Search Results` and the code context could enhance efficiency. Additionally, switching to the Tool Control Window(`ActiveWindowChanged | ToolControlWindow`) is present in cluster 8, although it is less prominent than the code editing and searching activities. Optimizing the workflow for accessing and using the Tool Control Window could also be considered.

From Table A.33, Table A.34, and Table A.35, the `ActiveWindowChanged | ControllerBrowser` → `ActiveWindowChanged | RapidEditorWindow` → `ExecutedCommand | RapidEditorShow` sequence is again central to cluster 9, highlighting the importance of code editing and controller management. Also switching to the Search Results window (`ActiveWindowChanged | SearchResults`), Graphic Window (`ActiveWindowChanged | GraphicWindow`) and Object Browser (`ActiveWindowChanged | ObjectBrowser`) are present. RobotStudio could offer more flexible workspace configurations to combine code editing, controller management, simulation viewing, and search functionalities. Keeping context when switching between these windows would also be beneficial.

In summary, in most clusters (0, 1, 3, 4, 5, 6, 8, 9, 10), users frequently switch between the Rapid Editor Window and the Controller Browser, which is consistent with the global n-gram results. Many clusters (1, 4, 5, 7) show users interacting with the simulation environment via the Graphic Window and Object/Element Browsers to visualize and inspect virtual robots. Clusters 6, 8, 9, and 10 indicate that users often search for specific information with `ActiveWindowChanged | SearchResults`. Clusters 1, 6, 7, and 8 show users working with Tool Control Window. Besides, a unique pattern involving `ActiveWindowChanged | SisAnalyzer.SisFileList` and `ActiveWindowChanged | SisAnalyzer.SummaryTable` is only present in cluster 2, though with low file coverage. Cluster 3 focuses mainly on switching between the Rapid Editor and Controller Browser. Based on these observations, potential improvements for RobotStudio include workspace optimization for common workflows, enhanced navigation and integration between key windows, improved contextual awareness when switching windows, better integration of search functionality, and a review of the accessibility and integration of special tools like SisAnalyzer. However, due to the nature of the provided data, it is difficult to find the precise buttons or specific areas that users click or visit. Therefore, the common workflow improvements and navigation enhancements are a bit unrealistic in practice.

4.4.3.2 Unfiltered N-gram Analysis Per Cluster

Same with filtered n-gram analysis per cluster, we mainly analyze the top 10 N-gram sequences for major clusters, including cluster 0, 2, 5, 8, and 9, because they have the top 5 file counts across all clusters.

From Table A.39, Table A.40 and Table A.41 in cluster 0, there is a high frequency of warnings (`UserLogMessage | Warning`) and errors (`UserLogMessage | Error`), application loading (`AddinLoading | I/O Engineering`, `AddinLoaded | I/O Engineering`, `AddinLoaded | Visual SafeMove 2`), repetitive network messages (`NetworkAdapterChange`), `UIThreadWatchdog | Warning`, and some GUI activity (`ActiveWindowChanged | GraphicWindow`). This suggests that users in this cluster may be encountering numerous issues or performing initial setup and loading tasks. Hence, it's crucial to investigate the root causes of the high volume of warnings and errors and provide more robust error handling and recovery mechanisms within RobotStudio. Besides, the repetitive network messages might indicate frequent configuration changes that could be streamlined. Improving the application loading process to ensure a smoother and faster startup would also enhance the user experience.

From Table A.45, Table A.46, Table A.47 in cluster 2, there is a high frequency of (`UserLogMessage | Warning`) and errors (`UserLogMessage | Error`), along with significant interaction between the Controller Browser (`ActiveWindowChanged | ControllerBrowser`) and the Rapid Editor Window (`ExecutedCommand | RapidEditorShow`). This may be because users are encountering errors while editing and simulating robot behavior. Therefore, efforts should be made to reduce the error rate during simulation and code editing, improving the clarity of error messages, and providing troubleshooting guides within RobotStudio. Besides, the repetitive switching between the Controller Browser and the Rapid Editor Window indicates a need to optimize the user interface for a smoother and more integrated workflow between simulation and code editing tasks.

From Table A.54, Table A.55 and Table A.56 in cluster 5, there is a mix of (`UserLogMessage | Warning`), errors (`UserLogMessage | Error`), and API calls (`ApiInfo`), along with Controller Browser (`ActiveWindowChanged | ControllerBrowser`) and Rapid Editor Window (`ExecutedCommand | RapidEditorShow`). This indicates that users are actively involved in code editing, controller management, and utilizing API functions. Actionable insights suggest a need to address the underlying causes of warnings and errors related to code editing, controller management, and API calls to improve stability. Secondly, the frequent interaction between the Controller Browser and Rapid Editor Window suggests that optimizing the UI design for seamless transitions and information flow between these two critical areas would be beneficial. Lastly, given the API usage, providing more user-friendly tools, documentation, and examples for API interaction within RobotStudio could enhance the developer experience.

From Table A.62, Table A.63, and Table A.64, cluster 8 shows a high occurrence of network adapter changes (`NetworkAdapterChange`) and general errors (`Error`), coupled with switching between the Controller

Browser (`ActiveWindowChanged | ControllerBrowser`) and the Rapid Editor (`ExecutedCommand | RapidEditorShow`). This may suggest users are encountering connectivity issues and errors during code editing and controller management. Actionable insights include providing better mechanisms for managing network adapter changes within RobotStudio. Besides, the interaction between the controller and editor highlights the potential for streamlining the workflow.

From Table A.65, Table A.66, and Table A.67, cluster 9 similarly has a high frequency of warnings (`UserLogMessage | Warning`), network adapter changes (`NetworkAdapterChange`), general errors, and some interaction between the controller and editor. The advice here would be to thoroughly investigate and address the root causes of the prevalent warnings and errors to improve the overall stability and reliability. Additionally, providing better resolution mechanisms for network adapter change issues. Finally, optimizing the user experience for code editing and controller interaction, similar to other clusters.

In conclusion, unfiltered n-gram analysis identifies key user interaction patterns in RobotStudio, highlighting areas for potential UI improvements and stability enhancements. There are several issues: prevalent warnings and errors; users frequently switch between code editing and controller management; window handling poses challenges. These issues present significant opportunities to enhance the user experience. Direct solutions include more robust error handling. Streamlined workflows between critical interfaces are, however, difficult to implement due to the nature of the provided data. Improved network stability is important. Better guidance for complex tasks is necessary. Addressing these areas can significantly improve RobotStudio.

5

Conclusions

5.1 Disscussion

In the background section, we introduced three research questions central to this thesis:

- **RQ1** : Can meaningful user behavior patterns be discovered from RobotStudio event logs?
- **RQ2** : Can user behavior patterns be used to generate realistic test cases?
- **RQ3** : Can meaningful user behavior patterns be discovered from different clusters of RobotStudio event logs?

After performing clustering of event sequences and conducting pattern mining through N-gram and Markov chain analysis, we reached the following conclusions:

In addressing **RQ1**, our methodology included filtering, re-encoding, and sequential pattern mining on the collected event logs. This approach successfully identified structured and repetitive user interaction patterns, exemplified by the frequent switching between UI elements such as `RapidEditorWindow` and `ControllerBrowser`. Such interactions mirror typical user workflows, notably the cycle of edit–verify–adjust. The prevalence of repeated commands, like `RapidEditorShow` within structured N-gram, underscores regular navigation habits and reflects consistent task-driven user behaviors. Although raw event logs contained substantial system-level noise, our systematic filtering enabled the isolation of user-centric events, demonstrating clearly that meaningful behavior patterns are indeed extractable. This provides valuable insights into authentic user interactions within RobotStudio.

For **RQ2**, our findings also yield a positive conclusion. The frequent N-gram sequences discovered constitute plausible and routinely executed interaction flows, thereby serving as excellent candidates for prioritized test case generation. Further, the constructed Markov transition model facilitates probability-weighted sampling of interaction paths, closely mirroring the actual statistical distribution of user behaviors. Particularly compelling are transitions with probabilities of 1, indicating deterministic sequences suitable for robust test assertions. Additionally, the identification of nodes with high in-degree within the interaction graph highlights crucial convergence points in the user interface, suggesting priority areas for rigorous testing.

For **RQ3**, our results reveal different user behavior patterns across clusters, whether filtered or unfiltered. Most clusters involve frequent switching between the Rapid Editor and Controller Browser, indicating a core workflow of code editing and controller management. Additional patterns include interaction with the simulation environment (Graphic and Element Browsers), searching, and tool management. Here are some suggestions for RobotStudio: enhance navigation and integrate between key windows, improve contextual awareness when switching windows, better integration of search functionality, and review the accessibility and integration of special tools. Unfiltered N-gram analysis further highlights the prevalence of warnings and errors, suggesting opportunities to improve UI stability and error handling.

However, the granularity of our log data predominantly restricts our insights to view-level transitions between distinct UI components. This limitation underscores the necessity for more detailed logging practices, such as capturing interactions at the intra-window level, including individual clicks, drag-and-drop operations, and field interactions. Such granular data could significantly enrich our understanding of user goals, challenges, and interface inefficiencies. Additionally, this project only contains the necessary information for data preprocessing, which may lose some background context for later experiments. Besides, 1749 RobotStudio log files may not generalize the overall user behavior patterns with the time limit.

Nonetheless, the project effectively establishes a practical framework for deriving and modelling user behavior from event logs, laying foundational groundwork for future advancements in log-based UX analytics and behavior-driven testing methodologies.

5.2 Future Work

Building upon the current findings, future research can be directed towards both short-term enhancements and longer-term expansions of the proposed methodology.

5.2.1 Short-Term Goals

With an additional three months of research, a key objective would be to translate the current analytical process into a more automated, tool-based workflow. This would involve developing a user-friendly tool that empowers developers to define filters based on parameters such as product versions, user segments, and specific timeframes. Such a tool would then automatically execute the pattern mining analyses, offering a more efficient and customizable way to extract insights into user behavior.

Furthermore, constructing an interactive graph visualization, potentially leveraging graph databases in conjunction with Markov chain principles, could significantly improve exploratory analyses. This would allow developers to dynamically navigate and investigate the relationships between different events more intuitively.

To increase confidence in the analytical outcomes, incorporating systematic validation methods for the identified patterns would be a valuable next step. Finally, exploring the application of more advanced modelling techniques, such as higher-order Markov

models or Hidden Markov Models (HMMs), could potentially enhance the precision of our behavioral models and enable the capture of more intricate user dynamics.

5.2.2 Long-Term Goals

First, it is fundamental to enhance the precision and completeness of RobotStudio log files. Currently, the log entries in the provided data are either too technically detailed, making them difficult for developers and researchers to interpret. Or the log entries are too insufficient and ambiguous to draw obvious observations. To address this issue, we propose comprehensively standardizing the RobotStudio event log entry format. This standardization should process all event types, including errors, warnings, and command executions, ensuring easier data preprocessing. By implementing a unified format, we can lay the groundwork for more effective later analysis and possible improvement.

Additionally, RobotStudio can aim for an enhanced user experience through an intelligent interface that predicts user needs based on the current task being run and historical usage. This can be achieved by developing a machine learning prediction paradigm based on richer usage log data. It is very beneficial for users to reduce manual adjustments.

Besides enriching log information, there are several critical enhancements for RobotStudio's functionality and user experience. First, streamlining workflows to achieve integration across core functions will enhance user efficiency. Moreover, a key long-term goal is to reduce user-encountered errors as much as possible. The implementation of user guidance systems, context-aware assistance, and error prediction mechanisms can achieve this. Furthermore, RobotStudio should develop a highly customizable environment, encouraging users to personalize the interface and extended functionality. This change will foster a more efficient and user-centric experience.

Bibliography

[1] M. Javaid, A. Haleem, R. P. Singh, and R. Suman, "Substantial capabilities of robotics in enhancing industry 4.0 implementation," *Cognitive Robotics*, vol. 1, pp. 58–75, 2021, ISSN: 26672413. DOI: 10.1016/j.cogr.2021.06.001. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2667241321000057 (visited on 04/10/2025).

[2] M. R. Diprasetya, S. Yuwono, M. Löppenberg, and A. Schwung, "Integration of ABB robot manipulators and robot operating system for industrial automation," in *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, 2023, pp. 1–7. DOI: 10.1109/INDIN51400.2023.10217964.

[3] ABB, *ABB Factsheet*, https://search.abb.com/library/Download.aspx?DocumentID=9AKK108466A5292&LanguageCode=en&DocumentPartId=&Action=Launch, Accessed: 2025-03-06, 2025.

[4] ABB Robotics, *RobotStudio Suite*, https://new.abb.com/products/robotics/software-and-digital/robotstudio, Accessed: 2025-02-20, 2025.

[5] P. Trott, P. V. D. Duin, and D. Hartmann, "Users as innovators? Exploring the limitations of user-driven innovation," *Prometheus*, vol. 31, no. 2, Jun. 1, 2013, ISSN: 0810-9028, 1470-1030. DOI: 10.1080/08109028.2013.818790. [Online]. Available: https://scienceopen.com/hosted-document?doi=10.1080/08109028.2013.818790 (visited on 04/10/2025).

[6] N. Kühl and G. Satzger, "Needmining: Designing digital support to elicit needs from social media," *CoRR*, vol. abs/2101.06146, 2021. arXiv: 2101.06146. [Online]. Available: https://arxiv.org/abs/2101.06146.

[7] A. Tkalich, E. Klotins, T. Sporsem, V. Stray, N. B. Moe, and A. Barbala, *User Feedback in Continuous Software Engineering: Revealing the State-of-Practice*, 2024. DOI: 10.48550/ARXIV.2410.07459. (visited on 04/10/2025).

[8] K. De Moor, K. Berte, L. De Marez, W. Joseph, T. Deryckere, and L. Martens, "User-driven innovation? Challenges of user involvement in future technology analysis," *Science and Public Policy*, vol. 37, no. 1, pp. 51–61, Feb. 2010, ISSN: 03023427, 14715430. DOI: 10.3152/030234210X484775. (visited on 04/10/2025).

[9] A. Vidal Tost, "Innovation management in ABB - IT innovation designed for customer experience," Ph.D. dissertation, UPC, Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, Departament d'Organització d'Empreses, 2010. [Online]. Available: http://hdl.handle.net/2099.1/9384.

[10] A. Augusto, R. Conforti, M. Dumas, *et al.*, *Automated Discovery of Process Models from Event Logs: Review and Benchmark*, 2017. DOI: 10.48550/ARXIV.1705.02288. (visited on 04/10/2025).

[11] A. Pecchia and S. Russo, "Detection of Software Failures through Event Logs: An Experimental Study," in *2012 IEEE 23rd International Symposium on Software Reliability Engineering*, Dallas, TX, USA: IEEE, Nov. 2012, pp. 31–40, ISBN: 978-1-4673-4638-2. DOI: 10.1109/ISSRE.2012.24. (visited on 04/10/2025).

[12] A. Pecchia, M. Cinque, G. Carrozza, and D. Cotroneo, "Industry Practices and Event Logging: Assessment of a Critical Software Development Process," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Florence, Italy: IEEE, May 2015, pp. 169–178, ISBN: 978-1-4799-1934-5. DOI: 10.1109/ICSE.2015.145. (visited on 04/10/2025).

[13] J. Zhu, S. He, J. Liu, *et al.*, *Tools and Benchmarks for Automated Log Parsing*, 2018. DOI: 10.48550/ARXIV.1811.03509. (visited on 04/10/2025).

[14] N. D. Khan, J. A. Khan, J. Li, T. Ullah, and Q. Zhao, "Mining software insights: Uncovering the frequently occurring issues in low-rating software applications," *PeerJ Computer Science*, vol. 10, e2115, Jul. 2024, ISSN: 2376-5992. DOI: 10.7717/peerj-cs.2115. (visited on 04/11/2025).

[15] L. Carvajal, A. M. Moreno, M.-I. Sanchez-Segura, and A. Seffah, "Usability through Software Design," *IEEE Transactions on Software Engineering*, vol. 39, no. 11, pp. 1582–1596, Nov. 2013, ISSN: 0098-5589, 1939-3520. DOI: 10.1109/TSE.2013.29. (visited on 04/11/2025).

[16] N. Nakamichi, K. Shima, M. Sakai, and K.-i. Matsumoto, "Detecting low usability web pages using quantitative data of users' behavior," in *Proceedings of the 28th International Conference on Software Engineering*, Shanghai China: ACM, May 2006, pp. 569–576, ISBN: 978-1-59593-375-1. DOI: 10.1145/1134285.1134365. (visited on 04/11/2025).

[17] J.-R. Rehse, L. Abb, G. Berg, C. Bormann, T. Kampik, and C. Warmuth, "User Behavior Mining: A Research Agenda," *Business & Information Systems Engineering*, vol. 66, no. 6, pp. 799–816, Dec. 2024, ISSN: 2363-7005, 1867-0202. DOI: 10.1007/s12599-023-00848-1. (visited on 04/11/2025).

[18] S. Winter, S. Wagner, and F. Deissenboeck, "A Comprehensive Model of Usability," 2016. DOI: 10.48550/ARXIV.1612.04598. (visited on 04/11/2025).

[19] J. M. P. Jeba, J. M. P. Jeba, M. Bhuvaneswari, *et al.*, "Extracting usage patterns from web server log," *null*, 2016. DOI: 10.1109/icghpc.2016.7508074.

[20] C. Liu, C. Liu, J. Zhang, *et al.*, "A two-layered framework for the discovery of software behavior: A case study," *IEICE Transactions on Information and Systems*, 2018. DOI: 10.1587/transinf.2017edp7027.

[21] Y. Tao, S. Guo, C. Shi, and D. Chu, "User behavior analysis by cross-domain log data fusion," *IEEE Access*, vol. 8, pp. 400–406, 2020. DOI: 10.1109/ACCESS.2019.2961769.

[22] J. Han, J. Han, M. Kamber, M. Kamber, J. Pei, and J. Pei, "Data mining : Concepts and techniques 3rd edition ed. 3," *https://hanj.cs.illinois.edu/bk3/*, 2011.

[23] B. Liu, B. Liu, L. Zhang, and L. Zhang, "A survey of opinion mining and sentiment analysis," *Mining Text Data*, 2012. DOI: 10.1007/978-1-4614-3223-4_13.

[24] Y. Xie, Y. Xie, V. V. Phoha, V. V. Phoha, and V. V. Phoha, "Web user clustering from access log using belief function," *International Conference on Knowledge Capture*, 2001. DOI: 10.1145/500737.500768.

[25] M. Deshpande and G. Karypis, "Selective markov models for predicting web page accesses," *ACM transactions on internet technology (TOIT)*, vol. 4, no. 2, pp. 163–184, 2004.

[26] F. Chierichetti, F. Chierichetti, R. Kumar, *et al.*, "Are web users really Markovian," *The Web Conference*, 2012. DOI: 10.1145/2187836.2187919.

[27] N. Gurumdimma, A. Jhumka, M. Liakata, E. Chuah, and J. Browne, "Towards detecting patterns in failure logs of large-scale distributed systems," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, IEEE, 2015, pp. 1052–1061.

[28] R. Vaarandi and M. Pihelgas, "Logcluster - a data clustering and pattern mining algorithm for event logs," in *2015 11th International Conference on Network and Service Management (CNSM)*, 2015, pp. 1–7. DOI: 10.1109/CNSM.2015.7367331.

[29] S. Thaler, S. Thaler, V. Menkonvski, M. Petković, V. Menkonvski, and M. Petkovic, "Towards a neural language model for signature extraction from forensic logs," 2017. DOI: 10.1109/isdfs.2017.7916497.

[30] X. Zhang, Z. Xu, X. Yong, *et al.*, "Robust log-based anomaly detection on unstable log data," *Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, 2019. DOI: 10.1145/3338906.3338931.

[31] C. Bertero, M. Roy, C. Sauvanaud, and G. Tredan, "Experience report: Log mining using natural language processing and application to anomaly detection," in *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, 2017, pp. 351–360. DOI: 10.1109/ISSRE.2017.43.

[32] D. Sun, J. He, H. Zhang, Z. Qi, H. Zheng, and X. Wang, *A LongFormer-based framework for accurate and efficient medical text summarization*, 2025. DOI: 10.48550/ARXIV.2503.06888. (visited on 04/13/2025).

[33] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979, ISSN: 00359254, 14679876. [Online]. Available: http://www.jstor.org/stable/2346830 (visited on 02/21/2025).

[34] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Pearson Education India, 2016.

[35] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[36] B. S. Everitt, "Unresolved problems in cluster analysis," *Biometrics*, pp. 169–181, 1979.

[37] M. R. Ackermann, J. Blömer, D. Kuntze, and C. Sohler, "Analysis of Agglomerative Clustering," *Algorithmica*, vol. 69, no. 1, pp. 184–215, May 2014, ISSN: 0178-4617, 1432-0541. DOI: 10.1007/s00453-012-9717-4. [Online]. Available:

http://link.springer.com/10.1007/s00453-012-9717-4 (visited on 05/14/2025).

[38] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007, ISSN: 0960-3174, 1573-1375. DOI: 10.1007/s11222-007-9033-z. (visited on 04/14/2025).

[39] D. A. Reynolds *et al.*, "Gaussian mixture models.," *Encyclopedia of biometrics*, vol. 741, no. 659-663, p. 3, 2009.

[40] M. E. Celebi, *Unsupervised Learning Algorithms*. Cham: Springer International Publishing AG, 2016, 1 p., ISBN: 978-3-319-24211-8.

[41] P. Leach, M. Mealling, and R. Salz, *Rfc 4122: A universally unique identifier (uuid) urn namespace*, 2005.

[42] V. S and J. R, "Text Mining: Open Source Tokenization Tools – An Analysis," *Advanced Computational Intelligence: An International Journal (ACII)*, vol. 3, no. 1, pp. 37–47, Jan. 2016, ISSN: 24543934. DOI: 10.5121/acii.2016.3104. (visited on 05/05/2025).

[43] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[44] R. L. Thorndike, "Who belongs in the family?" *Psychometrika*, vol. 18, no. 4, pp. 267–276, Dec. 1953, ISSN: 0033-3123, 1860-0980. DOI: 10.1007/BF02289263. (visited on 04/14/2025).

[45] T. T. Cai and R. Ma, "Theoretical foundations of t-sne for visualizing high-dimensional clustered data," *Journal of Machine Learning Research*, vol. 23, no. 301, pp. 1–54, 2022.

A

Appendix 1

Table A.1: Event Type Abbreviations with Descriptions

Abbreviation	**Full Form**	**Description**
AWC|REW	ActiveWindowChanged | RapidEditorWindow	Switching to or focusing on the Rapid Editor window for robot programming scripts.
AWC|CB	ActiveWindowChanged | ControllerBrowser	Switching to or focusing on the Controller Browser for managing robot controllers.
AWC|GW	ActiveWindowChanged | GraphicWindow	Switching to or focusing on the Graphic Window displaying graphical robot simulations.
AWC|EB	ActiveWindowChanged | ElementBrowser	Switching to or focusing on the Element Browser for elements within the robot project.
AWC|OB	ActiveWindowChanged | ObjectBrowser	Switching to or focusing on the Object Browser, showing project objects and components.
AWC|TCW	ActiveWindowChanged | ToolControlWindow	Switching to or focusing on the Tool Control window for managing robot tools.
AWC|TW	ActiveWindowChanged | ToolWindow	Switching to or focusing on the Tool window for managing robot tools.
AWC|SR	ActiveWindowChanged | SearchResults	Switching to or focusing on the Search Results window after performing a search.
AWC|RW	ActiveWindowChanged | RapidWatch	Switching to or focusing on the Rapid Watch window

Continued on next page

Abbreviation	Full Form	Description
AWC|SCEW	ActiveWindowChanged | SmartComponentEditor-Window	Switching to or focusing on the Smart Component Editor window
AWC|IOE_SE	ActiveWindowChanged | IOE_SignalEditor_	Switching to or focusing on the IOE Signal Editor window
AWC|IOE_C	ActiveWindowChanged | IOE_Commissioning	Switching to or focusing on the IOE Commissioning window
AWC|MM	ActiveWindowChanged | MechanismModeler	Switching to or focusing on the Mechanism Modeler window.
AWC|CEW	ActiveWindowChanged | CfgEditorWindow	Switching to or focusing on the Configuration Editor window.
AWC|CCE	ActiveWindowChanged | ControllerConfigura-tionEditor	Switching to or focusing on the Controller Configuration Editor.
EC|RAA	ExecutedCommand | RapidApplyAll	Command execution to apply all pending Rapid changes across the project.
AWC|O	ActiveWindowChanged | Output	Switching to or focusing on the Output window showing log/output messages.
AWC|APVF	ActiveWindowChanged | ArcWeld Path View Form	Switching to or focusing on the ArcWeld Path View window form.
AWC|PTWI	ActiveWindowChanged | PathView Tool Window ID1	Switching to or focusing on a specific PathView tool window (identified by ID1).
AWC|SA.SFL	ActiveWindowChanged | SisAnalyzer.SisFileList	Switching to or focusing on the SA.SF file list view within the SisAnalyzer module.
AWC|SA.ST	ActiveWindowChanged | SisAna-lyzer.SummaryTable	Switching to or focusing on the summary table view within the SisAnalyzer module.
EC|REGTD	ExecutedCommand | RapidEditorGoToDefini-tion	Executing the command to go to a definition in the Rapid Editor.
EC|VV	ExecutedCommand | ViewVisible	Command execution to toggle or make a view visible in the application.
EC|CRWA	ExecutedCommand | ControllerRe-leaseWriteAccess	Command to release write access to a controller

Continued on next page

Abbreviation	Full Form	Description
EC|RES	ExecutedCommand | RapidEditorShow	Executing command to show or refresh Rapid Editor, typically to edit robot code.
EC|AI	ExecutedCommand | AcquireImage	Executing command for acquiring or capturing images within the application.
EC|SS	ExecutedCommand | SimulationStep	Executing a simulation step command to advance robot simulation by one incremental step.
ULM|Err	UserLogMessage | Error	A logged error message triggered by the user or system.
ULM|Warn	UserLogMessage | Warning	A logged warning message indicating potential issues or cautionary notices.
UITW|Warn	UIThreadWatchdog | Warning	A warning indicating that the UI thread is experiencing slowdowns or unresponsiveness.
UITW|Err	UIThreadWatchdog | Error	An error indicating significant UI thread failures or unresponsiveness.
UITW|Start	UIThreadWatchdog | Started	Indicator that the UI watchdog thread has started monitoring the UI responsiveness.
NAC	NetworkAdapterChange	Notification of changes or updates in the network adapter status.
EXC|NRE	Exception | NullReferenceException	Occurrence of a runtime exception due to referencing a null object.
EXC|SFE	Exception | System.FormatException	Occurrence of a runtime exception caused by an invalid string format during type conversion.
API	ApiInfo	Event related to API calls or informational logging regarding API interactions.
EXC|RES	ExecutedCommand | RapidEditorShow	Command execution initiated for displaying the Rapid Editor window.
STL	StationLoaded	Event indicating that a robot station configuration has been loaded.
SVC	StartingVC	Starting the Visual Components or Virtual Controller component in the application.

Continued on next page

Abbreviation	**Full Form**	**Description**
VCA	VCAffinity	Event tracking thread/core affinity of the Virtual Controller.
ERR	Error	Generic or unknown error event in the system.
AA|IE	AddinAdded | I/O Engineering	Add-in registered for the I/O Engineering module.
AA|VS2	AddinAdded | Visual SafeMove 2	Add-in registered for the Visual SafeMove 2 module.
AL|IE	AddinLoading | I/O Engineering	Add-in loading process started for the I/O Engineering module.
AL|VS2	AddinLoading | Visual SafeMove 2	Add-in loading process started for the Visual SafeMove 2 module.
ALD|IE	AddinLoaded | I/O Engineering	Add-in successfully loaded for the I Engineering module.
ALD|VS2	AddinLoaded | Visual SafeMove 2	Add-in successfully loaded for the Visual SafeMove 2 module.
GFL	GraphicsFeatureLevel	Logging the graphics rendering feature level used by the application.

Table A.2: Top 10 6-gram by Total Occurrences (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn ×6	383	31919
2	Err ×6	54	10632
3	UITW|Warn ×6	183	7604
4	ULM|Err ×6	151	6064
5	EXC|NRE ×6	11	3769
6	NAC ×6	190	3762
7	AWC|REW ×6	106	2214
8	API ×6	48	2207
9	AWC|REW → EC|RES → AWC|CB → EXC|RES → AWC|REW → EC|RES	269	1616
10	EXC|RES → AWC|REW → EC|RES → AWC|CB → EXC|RES → AWC|REW	269	1614

Table A.3: Top 10 8-gram by Total Occurrences (Abbreviated)

1	ULM|Warn ×8	340	29552
2	Err ×8	50	10385
3	ULM|Err ×8	111	4783
4	UITW|Warn ×8	145	4634
5	EXC|NRE ×8	11	3695
6	NAC ×8	146	2742
7	API ×8	46	1763
8	AWC|REW ×8	69	1708
9	AWC|CB → EXC|RES → AWC|REW → EC|RES → AWC|CB → EXC|RES → AWC|REW → EC|RES	253	1511
10	UITW|Warn ×7 → UITW|Err	119	1254

Table A.4: Top 10 6-gram by Total Occurrences (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW ×6	115	3141
2	(AWC|REW → AWC|CB) ×3	46	1810
3	(AWC|CB → AWC|REW) ×3	43	1739
4	(AWC|CB → AWC|REW → EC|RES) ×2	270	1608
5	(AWC|GW → AWC|EB) ×3	57	941
6	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	190	905
7	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	180	862
8	(AWC|EB → AWC|GW) ×3	54	859
9	EC|AI ×6	2	511
10	(AWC|SR → AWC|REW) ×3	56	458

Table A.5: Top 10 8-gram by Total Occurrences (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW ×8	76	2718
2	(AWC|REW → AWC|CB) ×4	33	1599
3	(AWC|CB → AWC|REW) ×4	32	1547
4	(AWC|CB → AWC|REW → EC|RES → AWC|CB) ×2	173	814
5	(AWC|REW → EC|RES → AWC|CB → AWC|REW) ×2	169	802
6	(AWC|GW → AWC|EB) ×4	44	675
7	(AWC|EB → AWC|GW) ×4	45	632
8	(EC|RES → AWC|CB → AWC|REW → EC|RES) ×2	127	505
9	EC|AI ×8	2	477
10	EC|SS ×8	5	331

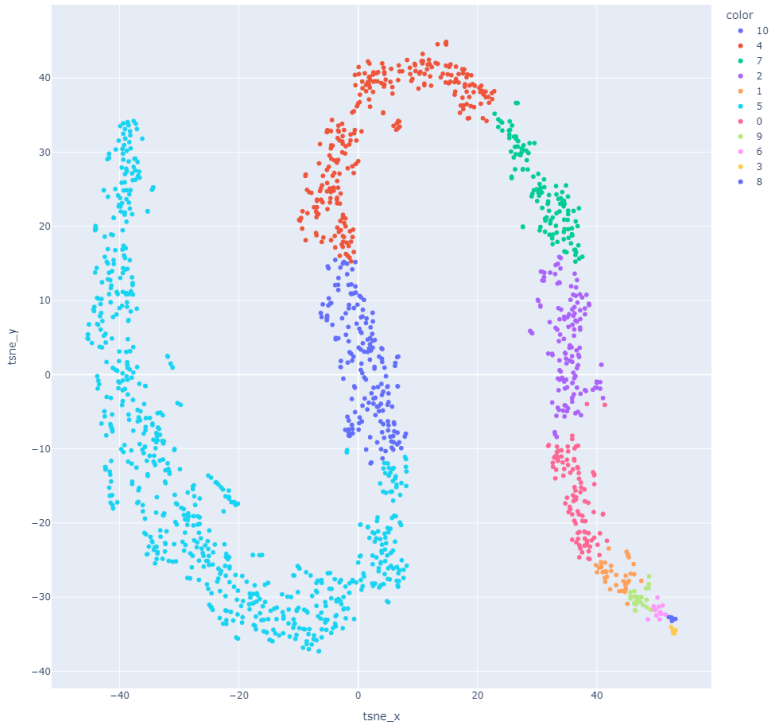

Figure A.1: T-SNE Visualization of Agglomerative Clustering Result

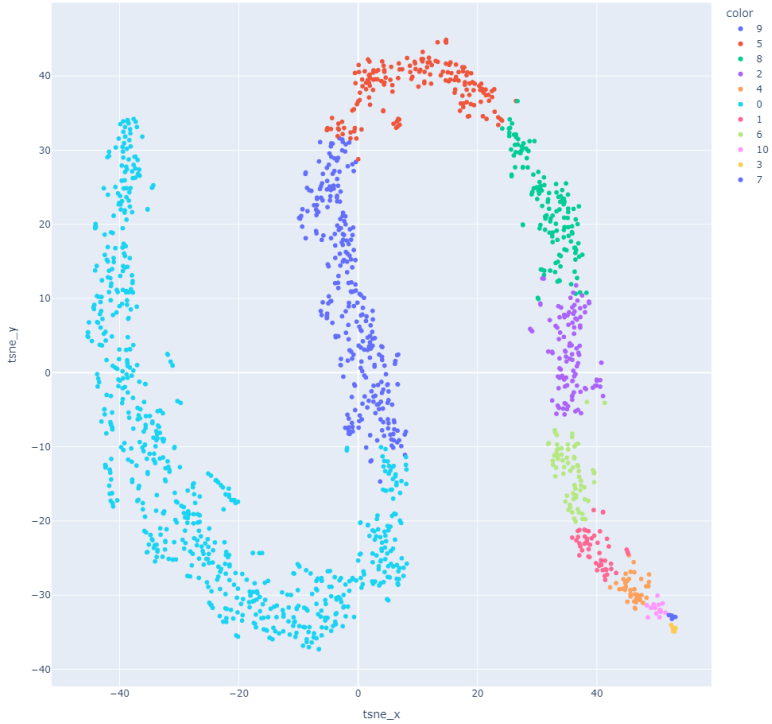

Figure A.2: T-SNE Visualization of GMM Clustering Result

Table A.6: Filtered Top 10 4-gram by Total Occurrences in Cluster 0 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → EC|RES → AWC|CB	18	31
2	AWC|REW → EC|RES → AWC|CB → AWC|REW	19	30
3	EC|RES → AWC|CB → AWC|REW → EC|RES	17	28
4	RTC|REW → AWC|CB → AWC|REW → EC|RES	22	24
5	AWC|GW → AWC|OB → AWC|GW → AWC|OB	11	21
6	AWC|OB → AWC|GW → AWC|OB → AWC|GW	9	17
7	AWC|GW → RTC|REW → AWC|CB → AWC|REW	14	14
8	AWC|CB → AWC|REW → EC|RES → AWC|REW	12	13
9	RTC|C → EC|CC → AWC|CB → AWC|REW	12	12
10	EC|CC → AWC|CB → AWC|REW → EC|RES	11	11

Table A.7: Filtered Top 10 6-gram by Total Occurrences in Cluster 0 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	16	27
2	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	6	12
3	AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB	6	10
4	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	4	10
5	RTC|REW → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	9	9
6	AWC|GW → RTC|REW → AWC|CB → AWC|REW → EC|RES → AWC|CB	8	8
7	AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW	5	8
8	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|REW	7	7
9	RTC|AI → AWC|PIW → AWC|DW → RTC|RL1 → EC|RLSB → AWC|O	6	6
10	AWC|REW → AWC|FT → AWC|REW → AWC|FT → AWC|REW → AWC|FT	1	6

Table A.8: Filtered Top 10 8-gram by Total Occurrences in Cluster 0 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	4	10
2	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	4	10
3	AWC|GW → RTC|REW → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	7	7
4	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	4	6
5	AWC|REW → AWC|FT → AWC|REW → AWC|FT → AWC|REW → AWC|FT → AWC|REW → AWC|FT	1	5
6	AWC|FT → AWC|REW → AWC|FT → AWC|REW → AWC|FT → AWC|REW → AWC|FT → AWC|REW	1	5
7	AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB	4	4
8	AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW	3	3
9	RTC|C → EC|CC → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	3	3
10	AWC|GW → RTC|AI → AWC|PIW → AWC|AIB → EC|PPL:DLW → AWC|GW → RTC|H → EC|VVC:GW	3	3

Table A.9: Filtered Top 10 4-gram by Total Occurrences in Cluster 1 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|GW → AWC|EB → AWC|GW → AWC|EB	20	473
2	AWC|EB → AWC|GW → AWC|EB → AWC|GW	18	412
3	AWC|CB → AWC|REW → EC|RES → AWC|CB	32	260
4	AWC|REW → EC|RES → AWC|CB → AWC|REW	30	254
5	EC|RES → AWC|CB → AWC|REW → EC|RES	28	233
6	AWC|TW → AWC|GW → AWC|TW → AWC|GW	1	223
7	AWC|GW → AWC|TW → AWC|GW → AWC|TW	1	222
8	AWC|CB → AWC|REW → AWC|CB → AWC|REW	12	221
9	AWC|REW → AWC|CB → AWC|REW → AWC|CB	11	221
10	AWC|GW → AWC|REW → AWC|GW → AWC|REW	7	127

Table A.10: Filtered Top 10 6-gram by Total Occurrences in Cluster 1 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	13	295
2	AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW	14	267
3	AWC|TW → AWC|GW → AWC|TW → AWC|GW → AWC|TW → AWC|GW	1	219
4	AWC|GW → AWC|TW → AWC|GW → AWC|TW → AWC|GW → AWC|TW	1	218
5	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	27	217
6	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	6	185
7	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	6	182
8	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	22	125
9	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	22	117
10	AWC|OB → AWC|TW → AWC|OB → AWC|TW → AWC|OB → AWC|TW	1	97

Table A.11: Filtered Top 10 8-gram by Total Occurrences in Cluster 1 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|TW → AWC|GW → AWC|TW → AWC|GW → AWC|TW → AWC|GW → AWC|TW → AWC|GW	1	216
2	AWC|GW → AWC|TW → AWC|GW → AWC|TW → AWC|GW → AWC|TW → AWC|GW → AWC|TW	1	215
3	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	12	204
4	AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW	12	184
5	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	5	166
6	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	5	163
7	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	22	111
8	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	19	109
9	AWC|OB → AWC|TW → AWC|OB → AWC|TW → AWC|OB → AWC|TW → AWC|OB → AWC|TW	1	96
10	AWC|TW → AWC|OB → AWC|TW → AWC|OB → AWC|TW → AWC|OB → AWC|TW → AWC|OB	1	96

Table A.12: Filtered Top 10 4-gram by Total Occurrences in Cluster 2 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|GW → AWC|EB → AWC|GW → AWC|EB	27	363
2	AWC|REW → AWC|CB → AWC|REW → AWC|CB	20	318
3	AWC|EB → AWC|GW → AWC|EB → AWC|GW	20	314
4	AWC|CB → AWC|REW → AWC|CB → AWC|REW	21	312
5	AWC|CB → AWC|REW → EC|REW → AWC|CB	48	293
6	AWC|REW → EC|REW → AWC|CB → AWC|REW	46	284
7	EC|REW → AWC|CB → AWC|REW → EC|REW	46	264
8	AWC|TCW → AWC|GW → AWC|TCW → AWC|GW	29	196
9	AWC|GW → AWC|TCW → AWC|GW → AWC|TCW	29	178
10	AWC|SA.SFL → AWC|SA.ST → AWC|SA.SFL → AWC|SA.ST	1	165

Table A.13: Filtered Top 10 6-gram by Total Occurrences in Cluster 2 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	12	265
2	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	15	264
3	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	11	260
4	AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW	14	256
5	AWC|CB → AWC|REW → EC|REW → AWC|CB → AWC|REW → EC|REW	41	245
6	AWC|SA.ST → AWC|SA.SF → AWC|SA.ST → AWC|SA.SF → AWC|SA.ST → AWC|SA.SF	1	162
7	AWC|SA.SF → AWC|SA.ST → AWC|SA.SF → AWC|SA.ST → AWC|SA.SF → AWC|SA.ST	1	161
8	AWC|REW → EC|REW → AWC|CB → AWC|REW → EC|REW → AWC|CB	29	146
9	EC|REW → AWC|CB → AWC|REW → EC|REW → AWC|CB → AWC|REW	28	135
10	AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW	23	104

Table A.14: Filtered Top 10 8-gram by Total Occurrences in Cluster 2 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	11	233
2	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	10	233
3	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	10	228
4	AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW	12	223
5	AWC|SA.ST → AWC|SA.SF → AWC|SA.ST → AWC|SA.SF → AWC|SA.ST → AWC|SA.SF → AWC|SA.ST → AWC|SA.SF	1	159
6	AWC|SA.SF → AWC|SA.ST → AWC|SA.SF → AWC|SA.ST → AWC|SA.SF → AWC|SA.ST → AWC|SA.SF → AWC|SA.ST	1	158
7	AWC|CB → AWC|REW → EC|REW → AWC|CB → AWC|REW → EC|REW → AWC|CB → AWC|REW	28	128
8	AWC|REW → EC|REW → AWC|CB → AWC|REW → EC|REW → AWC|CB → AWC|REW → EC|REW	28	127
9	EC|REW → AWC|CB → AWC|REW → EC|REW → AWC|CB → AWC|REW → EC|REW → AWC|CB	21	83
10	AWC|SEA → AWC|REW → AWC|SEA → AWC|REW → AWC|SEA → AWC|REW → AWC|SEA → AWC|REW	9	61

Table A.15: Filtered Top 10 4-gram by Total Occurrences in Cluster 3 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → AWC|CB → AWC|REW → AWC|CB	2	220
2	AWC|CB → AWC|REW → AWC|CB → AWC|REW	2	215
3	AWC|REW → AWC|RW → AWC|REW → AWC|RW	1	48
4	AWC|RW → AWC|REW → AWC|RW → AWC|REW	1	46
5	AWC|REW → EC|RES → AWC|CB → AWC|REW	4	36
6	EC|RES → AWC|CB → AWC|REW → EC|RES	4	35
7	AWC|CB → AWC|REW → EC|RES → AWC|CB	4	34
8	AWC|REW → AWC|CB → AWC|REW → EC|RES	4	24
9	AWC|CB → AWC|REW → EC|RES → AWC|REW	4	20
10	AWC|SCEW → AWC|TCW → AWC|SCEW → AWC|TCW	1	20

Table A.16: Filtered Top 10 6-gram by Total Occurrences in Cluster 3 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	2	206
2	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	2	203
3	AWC|REW → AWC|RW → AWC|REW → AWC|RW → AWC|REW → AWC|RW	1	45
4	AWC|RW → AWC|REW → AWC|RW → AWC|REW → AWC|RW → AWC|REW	1	44
5	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	4	32
6	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	4	16
7	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	4	16
8	AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C	1	14
9	AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE	1	12
10	AWC|SCEW → AWC|TCW → AWC|SCEW → AWC|TCW → AWC|SCEW → AWC|TCW	1	9

Table A.17: Filtered Top 10 8-gram by Total Occurrences in Cluster 3 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	1	195
2	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	1	192
3	AWC|REW → AWC|RW → AWC|REW → AWC|RW → AWC|REW → AWC|RW → AWC|REW → AWC|RW	1	43
4	AWC|RW → AWC|REW → AWC|RW → AWC|REW → AWC|RW → AWC|REW → AWC|RW → AWC|REW	1	42
5	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	4	16
6	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	4	15
7	AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C	1	12
8	AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE	1	11
9	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	4	7
10	EC|CRWA → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	1	6

Table A.18: Filtered Top 10 4-gram by Total Occurrences in Cluster 4 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → AWC|CB → AWC|REW → AWC|CB	9	900
2	AWC|CB → AWC|REW → AWC|CB → AWC|REW	8	848
3	AWC|GW → AWC|EB → AWC|GW → AWC|EB	13	314
4	AWC|GW → AWC|OB → AWC|GW → AWC|OB	14	267
5	AWC|EB → AWC|GW → AWC|EB → AWC|GW	13	255
6	AWC|OB → AWC|GW → AWC|OB → AWC|GW	14	212
7	AWC|CB → AWC|REW → EC|RES → AWC|CB	21	154
8	AWC|REW → EC|RES → AWC|CB → AWC|REW	20	154
9	EC|RES → AWC|CB → AWC|REW → EC|RES	20	145
10	AWC|SR → AWC|REW → AWC|SR → AWC|REW	9	122

Table A.19: Filtered Top 10 6-gram by Total Occurrences in Cluster 4 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	7	815
2	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	6	779
3	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	10	184
4	AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW	8	164
5	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	20	130
6	AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB	12	128
7	AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW	11	106
8	AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C	4	91
9	AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE	4	88
10	AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW	7	87

Table A.20: Filtered Top 10 8-gram by Total Occurrences in Cluster 4 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	6	752
2	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	5	723
3	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	8	134
4	AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW	7	128
5	AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C	2	82
6	AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE	2	80
7	AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB	9	69
8	AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW	5	63
9	AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW	8	57
10	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	14	56

Table A.21: Filtered Top 10 4-gram by File Coverage in Cluster 5 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → EC|RES → AWC|CB	36	114
2	AWC|REW → EC|RES → AWC|CB → AWC|REW	36	109
3	EC|RES → AWC|CB → AWC|REW → EC|RES	35	107
4	AWC|GW → AWC|OB → EC|VV → AWC|GW	23	49
5	AWC|GW → AWC|OB → AWC|GW → AWC|OB	17	46
6	AWC|TCW → AWC|GW → AWC|TCW → AWC|GW	14	42
7	AWC|REW → AWC|CB → AWC|REW → AWC|CB	7	39
8	AWC|CB → AWC|REW → EC|RES → AWC|REW	22	37
9	AWC|CB → AWC|REW → EC|RES → EC|RAA	19	36
10	AWC|OB → EC|VV → AWC|GW → AWC|OB	17	36

Table A.22: Filtered Top 10 6-gram by File Coverage in Cluster 5 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	32	98
2	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	23	59
3	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	17	54
4	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	4	23
5	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	3	21
6	AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW	7	20
7	AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW	7	20
8	AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW	4	18
9	AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR	4	17
10	AWC|OB → EC|VV → AWC|GW → AWC|OB → EC|VV → AWC|GW	7	16

Table A.23: Filtered Top 10 8-gram by File Coverage in Cluster 5 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	17	52
2	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	16	52
3	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	12	30
4	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	2	19
5	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	2	18
6	AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW	5	13
7	AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW	2	13
8	AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW	6	12
9	EC|RAA → AWC|O → AWC|REW → EC|RAA → AWC|O → AWC|REW → EC|RAA → AWC|O	3	11
10	AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR	3	11

Table A.24: Filtered Top 10 4-gram by Total Occurrences in Cluster 6 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → AWC|CB → AWC|REW → AWC|CB	12	303
2	AWC|CB → AWC|REW → EC|RES → AWC|CB	40	294
3	AWC|CB → AWC|REW → AWC|CB → AWC|REW	17	288
4	AWC|REW → EC|RES → AWC|CB → AWC|REW	36	285
5	EC|RES → AWC|CB → AWC|REW → EC|RES	34	269
6	AWC|CB → AWC|REW → EC|RES → AWC|REW	24	144
7	AWC|GW → AWC|OB → AWC|GW → AWC|OB	20	128
8	AWC|REW → AWC|CB → AWC|REW → EC|RES	30	126
9	AWC|SR → AWC|REW → AWC|SR → AWC|REW	13	112
10	AWC|GW → AWC|TCW → AWC|GW → AWC|TCW	20	104

Table A.25: Filtered Top 10 6-gram by Total Occurrences in Cluster 6 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	7	257
2	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	34	252
3	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	7	241
4	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	27	147
5	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	26	141
6	AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW	10	66
7	AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR	11	56
8	AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB	12	53
9	AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW	10	40
10	AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW	10	39

Table A.26: Filtered Top 10 8-gram by Total Occurrences in Cluster 6 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	5	230
2	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	5	218
3	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	26	139
4	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	24	133
5	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	19	92
6	AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW	9	42
7	AWC|CB → AWC|REW → EC|RES → AWC|REW → EC|REGTD:REW → AWC|CB → AWC|REW → EC|RES	8	33
8	AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR	8	31
9	(AWC|AWPVF → AWC|PTWI) ×4	2	24
10	(AWC|PTWI → AWC|AWPVF) ×4	2	23

Table A.27: Filtered Top 10 4-gram by Total Occurrences in Cluster 7 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|GW → AWC|EB → AWC|GW → AWC|EB	5	315
2	AWC|CB → AWC|REW → EC|RES → AWC|CB	4	254
3	EC|RES → AWC|CB → AWC|REW → EC|RES	4	252
4	AWC|REW → EC|RES → AWC|CB → AWC|REW	4	248
5	AWC|EB → AWC|GW → AWC|EB → AWC|GW	5	222
6	AWC|TCW → AWC|EB → AWC|TCW → AWC|EB	5	193
7	AWC|EB → AWC|TCW → AWC|EB → AWC|TCW	5	179
8	AWC|OB → AWC|GW → AWC|OB → AWC|GW	2	169
9	AWC|GW → AWC|TCW → AWC|GW → AWC|TCW	4	140
10	AWC|TCW → AWC|GW → AWC|TCW → AWC|GW	4	138

Table A.28: Filtered Top 10 6-gram by Total Occurrences in Cluster 7 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	4	243
2	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	5	128
3	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	3	123
4	AWC|TCW → AWC|EB → AWC|TCW → AWC|EB → AWC|TCW → AWC|EB	3	122
5	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	3	122
6	AWC|EB → AWC|TCW → AWC|EB → AWC|TCW → AWC|EB → AWC|TCW	3	120
7	AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW	5	106
8	AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW	2	86
9	AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW	4	81
10	AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW	4	79

Table A.29: Filtered Top 10 8-gram by Total Occurrences in Cluster 7 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	3	117
2	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	3	117
3	AWC|TCW → AWC|EB → AWC|TCW → AWC|EB → AWC|TCW → AWC|EB → AWC|TCW → AWC|EB	2	99
4	AWC|EB → AWC|TCW → AWC|EB → AWC|TCW → AWC|EB → AWC|TCW → AWC|EB → AWC|TCW	2	93
5	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	5	65
6	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	3	65
7	AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW	5	57
8	AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW	3	57
9	AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW	4	56
10	AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW	2	44

Table A.30: Filtered Top 10 4-gram by Total Occurrences in Cluster 8 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → EC|RES → AWC|CB	57	287
2	AWC|REW → EC|RES → AWC|CB → AWC|REW	56	266
3	EC|RES → AWC|CB → AWC|REW → EC|RES	52	257
4	AWC|REW → AWC|CB → AWC|REW → AWC|CB	16	169
5	AWC|CB → AWC|REW → AWC|CB → AWC|REW	15	147
6	AWC|SR → AWC|REW → AWC|SR → AWC|REW	17	123
7	AWC|REW → AWC|SR → AWC|REW → AWC|SR	13	103
8	AWC|TCW → AWC|GW → AWC|TCW → AWC|GW	21	101
9	AWC|REW → AWC|CB → AWC|REW → EC|RES	40	93
10	AWC|GW → AWC|TCW → AWC|GW → AWC|TCW	18	93

Table A.31: Filtered Top 10 6-gram by Total Occurrences in Cluster 8 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	49	237
2	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	35	150
3	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	33	132
4	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	10	130
5	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	7	113
6	AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW	12	89
7	AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR	12	77
8	AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW	10	57
9	AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW	9	55
10	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|REW	30	38

Table A.32: Filtered Top 10 8-gram by Total Occurrences in Cluster 8 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	33	129
2	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	31	126
3	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	6	109
4	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	6	99
5	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	23	87
6	AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW	12	67
7	AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR	10	57
8	AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW	8	37
9	AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW → AWC|TCW → AWC|GW	7	37
10	AWC|Output → AWC|REW → EC|RAA → AWC|Output → AWC|REW → EC|RAA → AWC|Output → AWC|REW	1	32

Table A.33: Filtered Top 10 4-gram by Total Occurrences in Cluster 9 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → EC|RES → AWC|CB	48	120
2	AWC|REW → EC|RES → AWC|CB → AWC|REW	43	111
3	EC|RES → AWC|CB → AWC|REW → EC|RES	40	99
4	AWC|SR → AWC|REW → AWC|SR → AWC|REW	9	41
5	RTC|REW → AWC|CB → AWC|REW → EC|RES	35	39
6	AWC|REW → AWC|SR → AWC|REW → AWC|SR	7	33
7	AWC|CB → AWC|REW → EC|RES → AWC|REW	25	29
8	AWC|GW → AWC|OB → EC|VV → AWC|GW	11	21
9	AWC|CB → AWC|CEW → EC|CCE → AWC|CB	11	20
10	AWC|GW → AWC|OB → AWC|GW → AWC|OB	8	18

Table A.34: Filtered Top 10 6-gram by Total Occurrences in Cluster 9 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	40	96
2	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	28	65
3	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	25	58
4	AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW	6	25
5	AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR	6	21
6	RTC|REW → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	16	16
7	AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW	1	14
8	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|REW	12	13
9	AWC|GW → AWC|OB → AWC|GW → AWC|OB → AWC|GW	1	13
10	AWC|CB → AWC|CEW → EC|CCE → AWC|CB → AWC|CEW → EC|CCE	6	11

Table A.35: Filtered Top 10 8-gram by Total Occurrences in Cluster 9 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	24	56
2	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	24	52
3	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	19	35
4	AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR	5	16
5	AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW	4	13
6	AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	1	13
7	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW	1	12
8	RTC|REW → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC	9	9
9	AWC|MM → AWC|GW → AWC|MM → AWC|GW → AWC|MM → AWC|GW → AWC|MM	1	8
10	AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C → AWC|IOE_SE → AWC|IOE_C	1	7

Table A.36: Filtered Top 10 4-gram by Total Occurrences in Cluster 10 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|REW → AWC|CB → AWC|REW → AWC|CB	5	187
2	AWC|CB → AWC|REW → AWC|CB → AWC|REW	5	172
3	AWC|GW → AWC|EB → EC|CTMC → AWC|GW	1	166
4	AWC|EB → EC|CTMC → AWC|GW → AWC|EB	1	163
5	EC|CTMC → AWC|GW → AWC|EB → EC|CTMC	1	152
6	AWC|SR → AWC|REW → AWC|SR → AWC|REW	2	90
7	AWC|CB → AWC|REW → EC|RES → AWC|CB	9	86
8	AWC|REW → EC|RES → AWC|CB → AWC|REW	8	81
9	EC|RES → AWC|CB → AWC|REW → EC|RES	8	74
10	AWC|REW → AWC|SR → AWC|REW → AWC|SR	2	74

Table A.37: Filtered Top 10 6-gram by Total Occurrences in Cluster 10 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|EB → EC|CTMC → AWC|GW → AWC|EB → EC|CTMC → AWC|GW	1	137
2	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	3	133
3	AWC|GW → AWC|EB → EC|CTMC → AWC|GW → AWC|EB → EC|CTMC	1	129
4	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	3	125
5	EC|CTMC → AWC|GW → AWC|EB → EC|CTMC → AWC|GW → AWC|EB	1	123
6	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	8	72
7	AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW	2	66
8	AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR	2	54
9	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB	5	39
10	EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	5	37

Table A.38: Filtered Top 10 8-gram by Total Occurrences in Cluster 10 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	AWC|EB → EC|CTMC → AWC|GW → AWC|EB → EC|CTMC → AWC|GW → AWC|EB → EC|CTMC	1	110
2	AWC|GW → AWC|EB → EC|CTMC → AWC|GW → AWC|EB → EC|CTMC → AWC|GW → AWC|EB	1	108
3	EC|CTMC → AWC|GW → AWC|EB → EC|CTMC → AWC|GW → AWC|EB → EC|CTMC → AWC|GW	1	105
4	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	3	101
5	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	3	95
6	AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW	2	50
7	AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR → AWC|REW → AWC|SR	2	41
8	AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW	5	37
9	AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES → AWC|CB → AWC|REW → EC|RES	5	33
10	AWC|FTEW → EC|FS → AWC|FTEW → AWC|FTEW → EC|FS → AWC|FTEW → AWC|FTEW → EC|FS	1	21

Table A.39: Unfiltered Top 10 4-grams by File Coverage in Cluster 0 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	96	970
2	AL|IE → ALD|IE → AL|VS2 → ALD|VS2	573	573
3	NAC → NAC → NAC → NAC	45	382
4	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	54	230
5	ALD|IE → AL|VS2 → ALD|VS2 → UITW|Start	193	193
6	ALD|IE → AL|VS2 → ALD|VS2 → GFL	181	181
7	AL|VS2 → ALD|VS2 → GFL → UITW|Warn	155	155
8	UITW|Start → AL|IE → ALD|IE → AL|VS2	140	140
9	ULM|Err → ULM|Err → ULM|Err → ULM|Err	29	136
10	STL → AWC|GW → SVC → VCA	129	130

Table A.40: Unfiltered Top 10 6-grams by File Coverage in Cluster 0 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	62	782
2	NAC → NAC → NAC → NAC → NAC → NAC	28	245
3	AL|IE → ALD|IE → AL|VS2 → ALD|VS2 → GFL → UITW|Warn	147	147
4	AA|VS2 → AL|IE → ALD|IE → AL|VS2 → ALD|VS2 → UITW|Start	117	117
5	AL|IE → ALD|IE → AL|VS2 → ALD|VS2 → UITW|Start → GFL	116	116
6	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	24	107
7	AA|VS2 → UITW|Start → AL|IE → ALD|IE → AL|VS2 → ALD|VS2	100	100
8	AA|IE → AA|VS2 → UITW|Start → AL|IE → ALD|IE → AL|VS2	98	98
9	UITW|Start → AL|IE → ALD|IE → AL|VS2 → ALD|VS2 → GFL	93	93
10	AA|IE → AA|VS2 → AL|IE → ALD|IE → AL|VS2 → ALD|VS2	90	90

Table A.41: Unfiltered Top 10 8-grams by File Coverage in Cluster 0 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	57	654
2	NAC → NAC → NAC → NAC → NAC → NAC → NAC → NAC	21	162
3	AA|IE → AA|VS2 → UITW|Start → AL|IE → ALD|IE → AL|VS2 → ALD|VS2 → GFL	64	64
4	AA|IE → AA|VS2 → AL|IE → ALD|IE → AL|VS2 → ALD|VS2 → UITW|Start → GFL	60	60
5	AA|VS2 → UITW|Start → AL|IE → ALD|IE → AL|VS2 → ALD|VS2 → GFL → UITW|Warn	57	57
6	ULM|Err → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	56	56
7	AA|VS2 → ULM|Err → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	50	50
8	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	15	49
9	AA|IE → AA|VS2 → ULM|Err → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	46	46
10	ERR → ERR → ERR → ERR → ERR → ERR → ERR → ERR	2	44

Table A.42: Unfiltered Top 10 4-grams by File Coverage in Cluster 1 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	29	5041
2	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	20	2255
3	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	3	771
4	AWC|CB → EC|RES → AWC|REW → EC|RES	38	663
5	ULM|Err → ULM|Err → ULM|Err → ULM|Err	21	592
6	AWC|GW → AWC|EB → AWC|GW → AWC|EB	20	459
7	API → API → API → API	4	448
8	AWC|EB → AWC|GW → AWC|EB → AWC|GW	18	401
9	AWC|REW → AWC|REW → AWC|REW → AWC|REW	22	320
10	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err	11	304

Table A.43: Unfiltered Top 10 6-grams by File Coverage in Cluster 1 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	22	4629
2	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	16	1570
3	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	3	757
4	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	16	438
5	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err	9	301
6	API → API → API → API → API → API	2	300
7	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err → UITW|Warn → UITW|Warn	9	297
8	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err → UITW|Warn	8	296
9	UITW|Warn → UITW|Warn → UITW|Err → UITW|Warn → UITW|Warn → UITW|Warn	9	293
10	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	13	289

Table A.44: Unfiltered Top 10 8-grams by File Coverage in Cluster 1 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	20	4359
2	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	13	925
3	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	3	743
4	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	13	349
5	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err	9	300
6	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err → UITW|Warn	8	294
7	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err → UITW|Warn → UITW|Warn	7	294
8	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err → UITW|Warn → UITW|Warn → UITW|Warn	7	291
9	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	8	287
10	UITW|Warn → UITW|Warn → UITW|Err → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	6	283

Table A.45: Unfiltered Top 10 4-grams by File Coverage in Cluster 2 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ERR → ERR → ERR → ERR	14	6662
2	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	66	3201
3	ULM|Err → ULM|Err → ULM|Err → ULM|Err	41	1229
4	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	4	1183
5	NAC → NAC → NAC → NAC	36	853
6	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	38	761
7	AWC|CB → EC|RES → AWC|REW → EC|RES	69	669
8	API → API → API → API	6	552
9	AWC|REW → AWC|REW → AWC|REW → AWC|REW	34	390
10	EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE	2	377

Table A.46: Unfiltered Top 10 6-grams by File Coverage in Cluster 2 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ERR → ERR → ERR → ERR → ERR → ERR	8	6541
2	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	50	2543
3	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	3	1174
4	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	23	951
5	NAC → NAC → NAC → NAC → NAC → NAC	28	553
6	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	27	497
7	API → API → API → API → API → API	6	435
8	EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE	2	332
9	AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES	40	239
10	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	12	238

Table A.47: Unfiltered Top 10 8-grams by File Coverage in Cluster 2 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ERR → ERR → ERR → ERR → ERR → ERR → ERR → ERR	8	6489
2	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	40	2194
3	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	3	1166
4	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	14	799
5	NAC → NAC → NAC → NAC → NAC → NAC → NAC → NAC	24	358
6	API → API → API → API → API → API → API → API	5	338
7	EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE	2	296
8	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	21	282
9	AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES	36	214
10	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	8	210

Table A.48: Unfiltered Top 10 4-grams by File Coverage in Cluster 3 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	6	14541
2	ULM|Err → ULM|Err → ULM|Err → ULM|Err	4	2306
3	ULM|Err → ULM|Err → ULM|Warn → ULM|Err	4	853
4	ULM|Err → ULM|Warn → ULM|Err → ULM|Warn	5	763
5	ULM|Warn → ULM|Err → ULM|Err → ULM|Warn	5	758
6	ULM|Warn → ULM|Err → ULM|Warn → ULM|Warn	5	739
7	ULM|Err → ULM|Warn → ULM|Err → ULM|Err	4	739
8	ULM|Err → ULM|Warn → ULM|Warn → ULM|Warn	6	710
9	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Err	5	706
10	ULM|Warn → ULM|Warn → ULM|Err → ULM|Err	5	671

Table A.49: Unfiltered Top 10 6-grams by File Coverage in Cluster 3 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	6	13595
2	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	3	1620
3	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	1	368
4	ULM|Err → ULM|Warn → ULM|Err → ULM|Warn → ULM|Err → ULM|Warn	4	354
5	ULM|Err → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	6	326
6	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Err	5	326
7	ULM|Warn → ULM|Err → ULM|Warn → ULM|Err → ULM|Warn → ULM|Err	4	322
8	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Warn → ULM|Err	3	298
9	ULM|Err → ULM|Err → ULM|Warn → ULM|Err → ULM|Err → ULM|Err	3	295
10	ULM|Err → ULM|Err → ULM|Err → ULM|Warn → ULM|Err → ULM|Err	4	291

Table A.50: Unfiltered Top 10 8-grams by File Coverage in Cluster 3 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	5	13035
2	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	3	1243
3	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	1	358
4	ULM|Err → ULM|Warn → ULM|Err → ULM|Warn → ULM|Err → ULM|Warn → ULM|Err → ULM|Warn	2	227
5	ULM|Warn → ULM|Err → ULM|Warn → ULM|Err → ULM|Warn → ULM|Err → ULM|Warn → ULM|Err	2	226
6	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Err	5	194
7	ULM|Err → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	6	193
8	ULM|Err → ULM|Err → ULM|Warn → ULM|Err → ULM|Err → ULM|Warn → ULM|Err → ULM|Err	3	159
9	ULM|Err → ULM|Err → ULM|Err → ULM|Warn → ULM|Err → ULM|Err → ULM|Warn → ULM|Err	3	153
10	ULM|Warn → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	3	150

Table A.51: Unfiltered Top 10 4-grams by File Coverage in Cluster 4 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	20	3061
2	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	19	2846
3	ULM|Err → ULM|Err → ULM|Err → ULM|Err	19	1584
4	AWC|REW → AWC|CB → AWC|REW → AWC|CB	9	791
5	AWC|CB → AWC|REW → AWC|CB → AWC|REW	7	708
6	ERR → ERR → ERR → ERR	5	548
7	ULM|Warn → API → ULM|Warn → API	2	523
8	API → ULM|Warn → API → ULM|Warn	2	520
9	AWC|CB → EC|RES → AWC|REW → EC|RES	30	491
10	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err	10	408

Table A.52: Unfiltered Top 10 6-grams by File Coverage in Cluster 4 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	14	2539
2	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	14	2120
3	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	8	1144
4	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	7	684
5	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	5	630
6	API → ULM|Warn → API → ULM|Warn → API → ULM|Warn	1	518
7	ULM|Warn → API → ULM|Warn → API → ULM|Warn → API	1	518
8	ERR → ERR → ERR → ERR → ERR → ERR	5	500
9	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err	10	406
10	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err → UITW|Warn	10	384

Table A.53: Unfiltered Top 10 8-grams by File Coverage in Cluster 4 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	11	2375
2	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	11	1236
3	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	7	864
4	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	4	616
5	AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW	4	595
6	API → ULM|Warn → API → ULM|Warn → API → ULM|Warn → API → ULM|Warn	1	517
7	ULM|Warn → API → ULM|Warn → API → ULM|Warn → API → ULM|Warn → API	1	517
8	ERR → ERR → ERR → ERR → ERR → ERR → ERR → ERR	3	465
9	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err	10	403
10	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err → UITW|Warn → UITW|Warn	10	383

Table A.54: Unfiltered Top 10 4-grams by File Coverage in Cluster 5 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	72	1720
2	ULM|Err → ULM|Err → ULM|Err → ULM|Err	37	980
3	API → API → API → API	14	821
4	ERR → ERR → ERR → ERR	10	789
5	NAC → NAC → NAC → NAC	42	552
6	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	48	443
7	AWC|CB → EC|RES → AWC|REW → EC|RES	72	276
8	EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE	6	215
9	AL|IE → ALD|IE → AL|VS2 → ALD|VS2	119	119
10	EC|RES → AWC|REW → EC|RES → AWC|CB	36	119

Table A.55: Unfiltered Top 10 6-grams by File Coverage in Cluster 5 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	53	1482
2	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	23	881
3	ERR → ERR → ERR → ERR → ERR → ERR	8	759
4	API → API → API → API → API → API	13	725
5	NAC → NAC → NAC → NAC → NAC → NAC	35	356
6	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	30	257
7	EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE	5	174
8	EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW	33	100
9	AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES	33	100
10	AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES	29	93

Table A.56: Unfiltered Top 10 8-grams by File Coverage in Cluster 5 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	48	1334
2	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	18	831
3	ERR → ERR → ERR → ERR → ERR → ERR → ERR → ERR	8	735
4	API → API → API → API → API → API → API → API	13	638
5	NAC → NAC → NAC → NAC → NAC → NAC → NAC → NAC	25	219
6	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	24	138
7	EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE	5	138
8	AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES	29	92
9	EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB	21	54
10	EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW	16	50

Table A.57: Unfiltered Top 10 4-grams by File Coverage in Cluster 6 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	31	2031
2	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	27	1517
3	ERR → ERR → ERR → ERR	12	1152
4	AWC|CB → EC|RES → AWC|REW → EC|RES	52	803
5	ULM|Err → ULM|Err → ULM|Err → ULM|Err	30	700
6	NAC → NAC → NAC → NAC	24	693
7	AWC|REW → AWC|REW → AWC|REW → AWC|REW	27	485
8	EC|RES → AWC|REW → EC|RES → AWC|CB	41	298
9	EC|RES → AWC|CB → EC|RES → AWC|REW	34	262
10	AWC|REW → EC|RES → AWC|CB → EC|RES	34	260

Table A.58: Unfiltered Top 10 6-grams by File Coverage in Cluster 6 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	28	1733
2	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	22	1136
3	ERR → ERR → ERR → ERR → ERR → ERR	8	1057
4	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	11	467
5	NAC → NAC → NAC → NAC → NAC → NAC	22	411
6	AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW	22	329
7	EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW	34	258
8	AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES	34	257
9	AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES	33	242
10	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	6	186

Table A.59: Unfiltered Top 10 8-grams by File Coverage in Cluster 6 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	24	1555
2	ERR → ERR → ERR → ERR → ERR → ERR → ERR → ERR	8	999
3	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	20	823
4	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	10	343
5	NAC → NAC → NAC → NAC → NAC → NAC → NAC → NAC	18	248
6	AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW	17	241
7	AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES	33	240
8	AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB → AWC|REW → AWC|CB	5	149
9	EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB	26	139
10	EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW	24	129

Table A.60: Unfiltered Top 10 4-grams by File Coverage in Cluster 7 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	5	983
2	AWC|CB → EC|RES → AWC|REW → EC|RES	5	502
3	AWC|REW → AWC|REW → AWC|REW → AWC|REW	2	400
4	ULM|Warn → EC|SS → EC|SS → ULM|Warn	1	353
5	AWC|GW → AWC|EB → AWC|GW → AWC|EB	5	283
6	EC|RES → AWC|CB → EC|RES → AWC|REW	4	244
7	EC|RES → AWC|REW → EC|RES → AWC|CB	4	238
8	EC|SS → EC|SS → ULM|Warn → ULM|Warn	2	232
9	AWC|REW → EC|RES → AWC|CB → EC|RES	4	228
10	ULM|Warn → ULM|Warn → EC|SS → EC|SS	2	200

Table A.61: Unfiltered Top 10 6-grams by File Coverage in Cluster 7 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	4	721
2	AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW	1	367
3	EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW	4	228
4	AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES	4	228
5	AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES	4	226
6	ULM|Warn → EC|SS → EC|SS → ULM|Warn → EC|SS → EC|SS	1	184
7	EC|SS → EC|SS → ULM|Warn → EC|SS → EC|SS → ULM|Warn	1	168
8	EC|SS → EC|SS → ULM|Warn → ULM|Warn → EC|SS → EC|SS	1	125
9	AWC|GW → AWC|EB → AWC|GW → AWC|EB → AWC|GW → AWC|EB	5	118
10	EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB	3	118

Table A.62: Unfiltered Top 10 4-grams by File Coverage in Cluster 8 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	NAC → NAC → NAC → NAC	42	1782
2	ERR → ERR → ERR → ERR	8	1260
3	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	29	1047
4	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	60	1043
5	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	2	765
6	AWC|CB → EC|RES → AWC|REW → EC|RES	83	599
7	ULM|Err → ULM|Err → ULM|Err → ULM|Err	31	285
8	EC|RES → AWC|REW → EC|RES → AWC|CB	57	276
9	EC|RES → AWC|CB → EC|RES → AWC|REW	50	236
10	AWC|REW → EC|RES → AWC|CB → EC|RES	47	230

Table A.63: Unfiltered Top 10 6-grams by File Coverage in Cluster 8 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	NAC → NAC → NAC → NAC → NAC → NAC	27	1409
2	ERR → ERR → ERR → ERR → ERR → ERR	7	1227
3	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	42	777
4	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	2	745
5	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	21	734
6	EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW	47	228
7	AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES	47	227
8	AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES	45	210
9	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	18	135
10	EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB	31	120

Table A.64: Unfiltered Top 10 8-grams by File Coverage in Cluster 8 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	NAC → NAC → NAC → NAC → NAC → NAC → NAC → NAC	20	1218
2	ERR → ERR → ERR → ERR → ERR → ERR → ERR → ERR	7	1200
3	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	2	725
4	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	33	647
5	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	16	465
6	AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES	45	208
7	EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB	30	118
8	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err	16	110
9	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err → UITW|Warn	15	109
10	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err → UITW|Warn → UITW|Warn	15	109

Table A.65: Unfiltered Top 10 4-grams by File Coverage in Cluster 9 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	115	2394
2	NAC → NAC → NAC → NAC	39	484
3	ERR → ERR → ERR → ERR	11	421
4	ULM|Err → ULM|Err → ULM|Err → ULM|Err	39	369
5	EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE	16	315
6	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	33	249
7	AWC|CB → EC|RES → AWC|REW → EC|RES	81	229
8	AL|IE → ALD|IE → AL|VS2 → ALD|VS2	203	203
9	API → API → API → API	10	182
10	ULM|Err → ULM|Warn → ULM|Warn → ULM|Warn	102	128

Table A.66: Unfiltered Top 10 6-grams by File Coverage in Cluster 9 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	97	2097
2	ERR → ERR → ERR → ERR → ERR → ERR	11	392
3	NAC → NAC → NAC → NAC → NAC → NAC	28	355
4	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	21	258
5	EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE	15	240
6	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	19	145
7	API → API → API → API → API → API	7	145
8	ULM|Err → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	92	97
9	EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW	39	90
10	AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES	38	89

Table A.67: Unfiltered Top 10 8-grams by File Coverage in Cluster 9 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	93	1877
2	ERR → ERR → ERR → ERR → ERR → ERR → ERR → ERR	11	365
3	NAC → NAC → NAC → NAC → NAC → NAC → NAC → NAC	20	269
4	ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err → ULM|Err	18	197
5	EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE → EXC|SFE	13	196
6	API → API → API → API → API → API → API → API	7	119
7	ULM|Err → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	90	94
8	AWC|CB → EC|RES → AWC|REW → EC|RES → AWC|CB → EC|RES → AWC|REW → EC|RES	37	85
9	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	17	83
10	AA|VS2 → ULM|Err → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	77	77

Table A.68: Unfiltered Top 10 4-grams by File Coverage in Cluster 10 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	8	1405
2	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	9	1205
3	AWC|REW → AWC|REW → AWC|REW → AWC|REW	7	1031
4	EC|AI → EC|AI → EC|AI → EC|AI	1	529
5	EC|AI → EC|AI → EC|AI → EC|AI	1	500
6	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	2	380
7	AWC|CB → EC|RES → AWC|REW → EC|RES	14	292
8	AWC|FTEW → AWC|FTEW → AWC|FTEW → AWC|FTEW	1	270
9	EC|RSI → EC|RSI → EC|RSI → EC|RSI	2	208
10	EC|RSI → EC|RSI → EC|RSI → EC|RSI	2	190

Table A.69: Unfiltered Top 10 6-grams by File Coverage in Cluster 10 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	5	1021
2	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	7	981
3	AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW	4	783
4	EC|AI → EC|AI → EC|AI → EC|AI → EC|AI → EC|AI	1	500
5	EC|AI → EC|AI → EC|AI → EC|AI → EC|AI → EC|AI	1	478
6	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	1	373
7	AWC|FTEW → AWC|FTEW → AWC|FTEW → AWC|FTEW → AWC|FTEW → AWC|FTEW	1	192
8	EC|RSI → EC|RSI → EC|RSI → EC|RSI → EC|RSI → EC|RSI	2	184
9	ICF|RSGFX → ICF|RSGFX → ICF|RSGFX → ICF|RSGFX → ICF|RSGFX → ICF|RSGFX	2	174
10	EC|RSI → EC|RSI → EC|RSI → EC|RSI → EC|RSI → EC|RSI	2	171

Table A.70: Unfiltered Top 10 8-grams by File Coverage in Cluster 10 (Abbreviated)

#	Sequence	File Coverage	Total Occurrences
1	ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn → ULM|Warn	5	928
2	AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW → AWC|REW	3	632
3	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn	6	608
4	EC|AI → EC|AI → EC|AI → EC|AI → EC|AI → EC|AI → EC|AI → EC|AI	1	478
5	EC|AI → EC|AI → EC|AI → EC|AI → EC|AI → EC|AI → EC|AI → EC|AI	1	461
6	EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE → EXC|NRE	1	367
7	EC|RSI → EC|RSI → EC|RSI → EC|RSI → EC|RSI → EC|RSI → EC|RSI → EC|RSI	2	167
8	ICF|RSGFX → ICF|RSGFX → ICF|RSGFX → ICF|RSGFX → ICF|RSGFX → ICF|RSGFX → ICF|RSGFX → ICF|RSGFX	2	162
9	UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Warn → UITW|Err	5	158
10	EC|RSI → EC|RSI → EC|RSI → EC|RSI → EC|RSI → EC|RSI → EC|RSI → EC|RSI	2	156

Table A.71: Event Pairs with Transition Probability = 1.0

Source Event (Count)	**Target Event (Count)**
EC|PathViewEditTarget (18)	AWC|ArcweldPathViewForm (392)
EC|ToolControlTearOffToolControlWindow (9)	AWC|ElementBrowser (12353)
IM|AWGunPSF25 (4)	EC|LibGalleryArcWelding (12)
IM|MyMechanism (5)	EC|FileImportLibrary (95)
EC|RapidEditorFindUnusedRefsREW (4)	AWC|REW (25376)
IM|BinzelID (3)	EC|LibGalleryArcWelding (12)
IM|IRBP$_L$300$_L$1250$_M$2009$_R$EV1$_0$1(2)	EC|LibGalleryABB (142)
IM|OmniCoreC90XT (3)	EC|LibGalleryOmniCore (3)
EC|LibGalleryOmniCore (3)	AWC|ObjectBrowser (9796)
EC|AddToConveyorObjectSource (15)	AWC|ObjectBrowser (9796)
IM|IRB1300$_{71}$40(7)	EC|LibGalleryABB (142)
IM|SpindleTool (2)	EC|LibGallery (27)
IM|IRB1300$_1$0$_1$15(5)	EC|LibGalleryABB (142)
EC|VSMEncapsulateUpperArm (2)	AWC|GraphicWindow (23733)
AWC|DeviceListWindow (2)	EC|FleetManagementNewDeviceList (2)
IM|IRB2600ID$_{82}$00(3)	EC|LibGalleryABB (142)
IM|EFGlassTrayPointer (2)	IM|EFPointer (6)
AWC|PhysicsMaterialManager (2)	EC|PhysicsMaterialManager (2)
IM|MyTool (23)	EC|LibGalleryTrainingObjects (55)
IM|TMillingTool (2)	EC|FileImportLibrary (95)
IM|IRB6660$_2$05$_1$90(2)	EC|LibGalleryABB (142)
AWC|ABBEducationTools (2)	EC|EduEducationTools (4)
AWC|TROB3 (4)	EC|EditInstructionTemplates (30)
AWC|GfxStatisticsWindow (2)	EC|GraphicsStatistics (4)
IM|IRB930$_2$2$_1$05$_3$0(2)	EC|LibGalleryABB (142)
EC|ABBTesterCloseButton (2)	AWC|PackageInstallationWindow (490)
EC|ControllerStartVC (2)	AWC|GraphicWindow (23733)
AWC|IOEAccessLevelEditorWindow (6)	AWC|IOECommissioning (418)

Continued on next page

Source Event (Count)	Target Event (Count)
AWC|IOEDeviceTrustLevelEditorWindow (6)	AWC|IOECommissioning (418)
EC|ShowCollisionGeometries (8)	AWC|GraphicWindow (23733)
EC|ChainConveyor (3)	AWC|ToolControlWindow (9819)
EC|VacuumTool (3)	AWC|ToolControlWindow (9819)
EC|PathViewEditJointTarget (3)	AWC|ArcweldPathViewForm (392)
IM|IRB2600ID$_1$5$_1$85(2)	EC|LibGalleryABB (142)
IM|T300L (2)	EC|FileImportLibrary (95)
IM|GripperDATN (2)	EC|FileImportLibrary (95)
EC|RapidEditorGoToVisualization (2)	AWC|ControllerBrowser (12006)
EC|EditFindFileTextEditorWindow (4)	AWC|SearchResults (1911)
EC|TubeStudioBtnWeldPath (4)	AWC|WeldTree (87)
EC|TubeStudioBtnOverview (2)	AWC|WeldTree (87)
IM|BinzelWH455D (4)	EC|LibGalleryArcWelding (12)
EC|PathShowProgrammedZones (4)	AWC|GraphicWindow (23733)
IM|IRB760$_4$50$_3$18(2)	IM|MU100$_0$1(4)
EC|Create3DPyramid (2)	AWC|GraphicWindow (23733)
EC|IOEToolAddInManageEdsFiles (2)	AWC|IOEPropertiesWindow (103)
RTC|SampleAddin1Tab1 (2)	EC|SampleAddin1StartButton (2)
AWC|RapidPathGraphicWindow/... (2)	AWC|REW (25376)
AWC|Help (2)	AWC|Results (108)
EC|ToolEditJointTarget (2)	AWC|ElementBrowser (12353)
EC|SisAnalyzerOpen (2)	AWC|SisAnalyzerFileList (176)
IM|IRB5500ProArmA130017207axis (2)	IM|IRB550027ABase (2)
EC|RapidCompareModuleWithEditor (2)	AWC|REW (25376)
VR|ButtonVPaint350 (2)	VR|InputModeVrSpeedModeDeactivate (5)
IM|Torch (3)	EC|FileImportLibrary (95)
IM|PJ58240702 (2)	EC|FileImportLibrary (95)
IM|PJ582 (2)	EC|FileImportLibrary (95)
EC|OpenRelation (2)	AWC|ControllerBrowser (12006)