Log mining to re-construct system behavior: an exploratory study on a large telescope system

Michele Pettinato, Free University of Bozen-Bolzano, Italy Email: michelepettinato95@gmail.com
Juan Pablo Gil, ALMA observatory, Chile
Email: Juan.Gil@alma.cl
Patricio Galeas, Universidad de la Frontera, Chile
Email: patricio.galeas@ufrontera.cl
Barbara Russo, Free University of Bozen-Bolzano, Italy
Email: barbara.russo@unibz.it

Abstract

Context: A large amount of information about system behavior is stored in logs that record system changes. Such information can be exploited to discover anomalies of a system and the operations that cause them. Given their large size, manual inspection of logs is hard and infeasible in a desired timeframe (e.g., real-time), especially for critical systems.
*Objective:*This study proposes a semi-automated method for reconstructing sequences of tasks of a system, revealing system anomalies, and associating tasks and anomalies to code components.
Method: The proposed approach uses unsupervised machine learning (Latent Dirichlet Allocation) to discover latent topics in messages of log events and introduces a novel technique based on pattern recognition to derive the semantic of such topics (topic labelling). The approach has been applied to the big data generated by the ALMA telescope system consisting of more than 2,000 log events collected in about five hours of telescope operation.
Results: With the application of our approach to such data, we were able to model the behavior of the telescope over 16 different observations. We found five different behavior models and three different types of errors. We use the models to interpret each error and discuss its cause.
Conclusions: With this work, we have also been able to discuss some of the known challenges in log mining. The experience we gather has been then summarized in lessons learned.

Keywords: Log mining, Latent Dirichlet Allocation, Text processing, System behavior

1. Introduction

System behavior is determined by a complex interplay of technical and non-technical factors in an operational environment, (i.e., a set of conditions under which the system operates). Low performance or unscheduled downtime can carry huge costs that seriously concern system managers and engineers. Therefore, understanding and, eventually, forecasting system behavior is of paramount importance, especially in critical systems that must be 24/7 operational.

System logs consist of events that keep track of system execution. *Log events* are automatically generated by the logging service of a system and are typically archived as a semi-structured short text (*event text*). Log mining is the process to discover patterns in log events. Such patterns have been used to model or predict system's behavior (Russo et al., 2015; Fronza et al., 2013; Mariani and Pastore, 2008; Goldstein et al., 2017; Chuah et al., 2010). Although there are several proposals to standardize the format of the event (Jiang et al., 2008; Russo et al., 2015; Salfner et al., 2006), no specific standard has been extensively adopted and mining logs becomes a sort of art with the following log mining known challenges:

Short texts. The unstructured part of the log text (i.e., *log message*) may be very short (e.g., one / two lines) and may include domain terms (e.g., acronyms). Thus, automated text analysis on individual log texts may be not that efficient (Zou and Song, 2016).

Context investigation. Log data is big and distributed. Data can be generated every millisecond and triggered by any application running in the system. As such, log mining requires a deep understanding of the system context and careful data preprocessing (Russo et al., 2015).

Sequences of events. System behavior (e.g., system operations) can be encapsulated in more than one log event. Therefore, log mining recently focuses on sequences of log events and mining logs requires context analysis to define and identify such sequences (Russo et al., 2015; Fronza et al., 2013) and the involvement of domain experts is of paramount importance (Russo et al., 2015).

Observation time window. Over the course of a system's lifetime, anything from software upgrades to minor configuration changes can drastically alter the meaning or character of the logs. Thus, a system must be observed in a period in which no

specific upgrade, test, or drastic change happen (Oliner et al., 2012).

Fully-automated log mining. Over the years, several tools have been developed to support log mining (e.g., (Rouillard, 2004), (Mariani and Pastore, 2008), although no fully-automated method is yet available and most of the mining effort is still manual and left to researchers, who may have a limited knowledge of the system and the environment in which it operates (Cinque et al., 2013).

Fault localization. Existing techniques of log mining provide useful insights about the possible locations of faults although developers must still put a significant effort to exactly identify the faults to be fixed (Gazzola et al., 2019; Salfner et al., 2006).

In this study, we illustrate the case of a system orchestrating the Atacama Large Millimeter Array telescope (ALMA)[1]. ALMA is the largest astronomical project in existence. It consists of a single telescope of revolutionary design, composed of 66 high precision antennas. Such system is compounded of 200 computers controlling the 66 antennas positioned in the Atacama desert in Chile at 5000 meters of altitude. The system is operational 24/7 and its logging service generates logs each millisecond for a total of up to 30 GB per day (log database size is measured in Terabytes) from six different subsystems triggered by over twenty heterogeneous teams at twelve participating institutes over the world (Gil et al., 2016). As such, mining logs of the ALMA system poses additional challenges, that are specific to its context, but that may incur in other similar large / complex / critical systems.

Log mining at ALMA (Fig. 3) is performed by the software engineers with the goal of finding the most 'optimal' query to interrogate the log dataset and reconstruct the cause of an error by visualizing log events time series that occur in a time window that incorporates the timestamp of the error reported in the ticket. All activities are performed manually and the 'optimum' is defined and reached when experts decide so, Table 3. Thus, *automatizing the procedure is a real need* and poses specific challenges of log mining:

Incomplete information. Deciphering the information in log events might not be simple. One reason concerns the user interface used to include the event description. Such interface may not incorporate all relevant information (Cinque et al., 2013) or may not be resilient to change with the same pace of the ALMA system. As result, some events may carry incomplete information about the system operations.

Identical timestamp. Log events can also occur with a precision lower than a millisecond and the logging system may not be able to distinguish one from the other. Thus, logs can be reported with identical timestamp and their order of occurrence is therefore lost. Thus, it is not completely straightforward to follow the logic of system just by looking at the individual logs sequentially and it becomes important to analyze them into clusters.

1.1. Research goal and contributions

The goal of the study is to design an approach to *reconstruct from system logs the (anomalous) behavior of a system as sequences of system's tasks and identify the low-level software components (i.e., files and methods) involved in the tasks.*

Once a system's anomaly is detected, maintainers can automatically associate telescope's tasks and code components to such misbehavior and avoid the manual, complex analysis of big sets of individual log events like the one described in Fig. 3 for the ALMA operations. To achieve this goal, we propose a semi-automated approach that combines existing machine learning method for log mining, the *Latent Dirichlet Allocation* (LDA), with pattern recognition for topic labelling. LDA is one of the most popular tools for text mining used to discover a hidden thematic structure (i.e., *latent topics*) from collections of textual documents (Blei et al., 2003). LDA is trained on a set of textual documents and produces a set of latent topics as bags of words extracted from the documents and a model that gives the probability for a new textual document to belong to any of such topics (i.e., *posterior probability*). We use LDA to discover the tasks of the ALMA telescope as latent topics of documents defined by sets of event messages. Therefore, topics are represented as bags of words of the ALMA log vocabulary. The process to associate a semantic to bags of word is called *topic labelling*. Such process is typically performed manually by the domain experts that interpret the words of a topic within the domain context (Layman et al., 2016). The semantic expressed by the bags of words might not be always enough to distinguish one topic from another. Thus, in our approach, the information associated to each topic is enriched: each topic is represented as a set of patterns of sequences of event messages recurring over documents of the ALMA corpus. The greater semantic of such patterns may better guide experts and researchers to find suitable labels than individual messages or words. As illustrative example, we applied our overall method to a sample of logs of the ALMA system and reflect on the lessons learned and how we overcome some of the challenges in log mining.

Summarizing, the contributions of this work are the following:

- An iterative method to identify the number of latent topics in log messages based on the coherency and independency of topics as sets of most relevant messages.
- A novel approach to label latent topics in log messages that exploits the natural time-ordering of sequences of log events and use pattern recognition on such sequences. By reading patterns of sequences of messages instead of bags of words researchers can better determine the semantic of a topic and associate it a label.
- A novel approach to model system's behavior as set of system's tasks by means of latent topics in sequences of log messages. By leveraging the information carried by log events, the model can also be used to localize the software classes and methods that are involved in each task. This is particularly useful for testers to trace in software the cause of system misbehavior.

[1] http://www.almaobservatory.org

– An application of the approach to the real case study of the complex system orchestrating the ALMA telescope. With the proposed model of system behavior, we are also able to describe the system's errors and their cause that would otherwise have been hidden behind the highly technical language of the logs.

– A final reflection on the challenges in log mining existing in literature and arising from the specific context of research.

The paper is organized as follows. We discuss the related work in Section 2. In Section 4, we introduce the research questions and illustrate the context analysis and the study samples. The approach is described in Section 5. In Section 6, we answer the research questions and we summaries the lesson learned and the threats to validity in Section 7 and 8 respectively. Finally, we conclude in Section 9.

2. Related Work

Three major research areas are relevant for the present study: 1) log mining, 2) topic modeling, and 3) topic labelling. In the following, we review the relevant literature according to these major perspectives.

2.1. Log mining

Oliner et al. (Oliner et al., 2012), overviewed some of the most common applications of log mining. According to the authors, log mining is usually performed to understand system performance. For example, system logs have been extensively used in diagnosing faults, scheduling applications and services of hardware (Featherstun and Fulp, 2010), or in dependable computing, and in computer networks management and monitoring (Fu and Xu, 2010). In software engineering, system logs have been employed to model the workflow design of activity processes (van der Aalst et al., 2003). In other cases, they have been used to identify abnormal system behavior against a system operational profile, i.e., the system expected behavior (Oliner and Stearley, 2007). Unfortunately, such operational profile is often not accessible without disclosing protected information. In addition, the operation profile is not always unique as it can evolve with the context.

Given the large amount of data that logging services can, in principle, produce, one of the major challenges is mining logs automatically. Mariani and Pastore (Mariani and Pastore, 2008) group automatic techniques supporting log mining into three major classes: specification-based techniques, expert systems and heuristic based techniques.

Specification-based techniques match events in log files with formal specifications that describe legal event sequences and the detected anomalies for verification (Andrews and Zhang, 2003). Such techniques reveal deviations from specifications, but they require the maintenance of complete and consistent specifications.

Expert systems use expert knowledge databases to describe the events that are commonly related to system errors. As such, methods of log mining are very tight to the single context (e.g., querying databases with regular expressions) as the databases are very connected to a specific operational environment (Simon and Malcolm, 2006). Thus, maintaining methods and databases is very expensive.

Heuristic-based approaches model event sequences with supervised and unsupervised machine learners (Gordon, 1999). Training supervised learners require the knowledge of behavior categories (e.g., error or regular behavior) (Russo et al., 2015) or generation of normal and anomalous behaviors in a controlled manner (e.g., (Bertero et al., 2017; Goldstein et al., 2017) whereas unsupervised algorithms, which identify errors as those that do not belong to any cluster of events or event sequences, require expert validation on the number of clusters and / or the found errors (Mariani and Pastore, 2008). In this work, we adopted the Heuristic-based approach with unsupervised machine learner (i.e., LDA) and provided a method to estimate the number of clusters needed by focusing on the semantic coherence of the events in clusters.

2.2. Topic modeling

Topic modeling is a text-mining approach for discovering hidden semantic structures in a text body. Recent research in software reliability has applied it to mine software issues for different purposes: from the detection of bug duplicates in bug reports (Nguyen et al., 2012) to triaging bug reports to developers (Xia et al., 2017). (Layman et al., 2016) applied topic modeling to bug reports at NASA to identify common problem across NASA missions whereas Damevski et al. (2016) modeled logs on IDE interactions.

The possibility to apply topic modeling relies on how difficult it is to adapt the modeling technique for natural language text to log data. Log events tracing the execution of a system are likely to be repetitive and predictable and, as such, follow some kind of naturalness like in natural language (Hindle et al., 2012). Thus, text processing techniques can potentially be exploited for log mining (Damevski et al., 2018). A good text processing model captures such regularity. Models based on individual word frequency (e.g., TF-IDF) have known limitations when applied to log text: they assume that the counts of different words provide independent evidence of similarity and they make no use of semantic similarities between words. Thus, these models do not capture position in text, semantic, co-occurrence in different documents etc. To incorporate context in modeling (e.g., words' co-occurrence), other approaches based on frequency and similarity of n-grams (e.g., sequence of words of length n) have been proposed in text processing, Guille and Favre (2014). In log mining though, system's tasks that compound systems' behavior are typically derived by aggregating the information carried by multiple messages, whose length and number are not known and can vary from tasks to task. Thus, approaches based on similarity of (equal-size) n-grams may not provide good model for logs.

Unsupervised classification as clustering (e.g., clustering) is another option that have some limitations when applied to logs. For example, k-means clustering and LDA are unsupervised

learning algorithms, where the user needs to decide a priori the parameter k, respectively the number of clusters and the number of topics. If both are applied to assign k topics to a set of N documents, the key difference is that k-means partitions the N documents in k disjoint clusters whereas LDA characterizes each document as a set of topics each with a given probability (e.g. Document D belongs for 60% to Topic A, 30% to topic B and 10% to topic E). This characterization helps reconstruct the documents as a set of topics as for the goal of this research (Section 1.1).

LDA and Hidden Markov Models are the most popular techniques recently used for log mining (Layman et al., 2016; Nguyen et al., 2012; Gadler et al., 2017; Damevski et al., 2016; Damevski et al., 2018; Khodabandelou et al., 2014). In this study, we use LDA on log event messages. One of the major challenges in applying LDA is the size and the richness of the vocabulary extracted from the corpus of text bodies. For example, an LDA analysis on short messages like tweets may be inefficient in isolating topics (Zou and Song, 2016).

Inefficiency in identifying topics in short messages is common to topic modeling techniques. To increase the efficiency of such techniques, Guille and Favre (2014) leverage the creation frequency of dynamic links that users insert in tweets to detect important events and estimate the magnitude of their impact over the crowd. To specifically increase LDA performance, a recent solution is to use a pooling schema improvement strategy by aggregating short messages before using them for LDA analysis (Mehrotra et al., 2013; Zou and Song, 2016; Damevski et al., 2018). Mehrotra et al. (2013) compared several schemas on tweets messages, including author-wise pooling (pooling tweets according to authors) and temporal pooling (pooling gathering all tweets posted within the same hour to capture the case when a major event occurs) whereas Damevski et al. (2016) pooled sequences of debug commands of debug sessions to build LDA documents.

In this study, we used the temporal pooling schema to aggregate event messages that belong to the same system's behavior (i.e., based on the time stamps between two begin events). As such, an LDA document is a text compounded by all such event messages. In our case, the pooling schema improvement technique enriches the vocabulary of the machine learner by integrating terms coming from messages related to the same task (i.e., the event sequence) of the telescope.

2.3. Topic labelling

In LDA, a topic is essentially a cluster of words of a given vocabulary that are ranked by the probability to belong to such cluster. Therefore, labelling a topic means associating to each of such clusters a representative concept (Lau et al., 2011). The typical approach for labelling topics in literature is analyzing the distribution of words in each topic with the help of domain experts (Layman et al., 2016; Hindle et al., 20). Recently, automated approaches parse online, textual references (e.g., Wikipedia) to derive such candidate labels and use supervised vector machines to associate labels to topics (Lau et al., 2011; Jiang et al., 2010). In this study, we propose an approach that uses information contained in patterns of message sequences to retrieve candidate labels from manuals and validate such candidates with domain experts.

3. Topic modeling and Latent Dirichlet Allocation

A topic model is a statistical algorithm for text mining used to discover the hidden thematic structure (i.e., latent topics) of a collection of documents.

In this work, the use of topic models is motivated by their capability to reduce dimensionality, which may be useful to raise the level of abstraction in logs from low-level event messages to higher level system's tasks. Specifically, documents are first build to represent the behavior of a system and then topic modeling is used to discover the tasks compounding such behavior.

LDA is one of the most popular method for topic modeling (Blei et al., 2003; Jiang et al., 2016; Zou and Song, 2016; Hindle et al., 20; Layman et al., 2016). An LDA model is a Bayesian generative model for discrete data where topics are assumed to be uncorrelated. It models a collection of text documents assuming that the words of each document arise from uncorrelated latent topics, each of which is characterized by a distribution over words of a given vocabulary (Blei et al., 2003). In LDA, a *word w* is the basic unit of discrete data, a *document d* is a bag of words, and a *corpus* is a collection of documents. In a *topic model* of a corpus, documents are represented as random mixtures over latent *topic*, where each topic is a distribution over all words of a given vocabulary V. Random mixture implies that the population of documents has multiple subpopulations and a word in a sample is assigned to many subpopulations that are indexed by using topics. Fig. 1 illustrates the model in terms of random and observed variables and their dependency structure. An LDA model is therefore generated from two prior probability Dirichlet distributions:

- $\theta \sim$ Dirichlet(α): representing the topic distribution over documents with hyper-parameter α,
- $\eta \sim$ Dirichlet(β): representing the word distribution over topics with hyper-parameter β.

Words in documents are described by a matrix expressing the probability of selecting a particular word when sampling a particular document. The model estimated by LDA consists of two matrices that "split" the association word-document into word-topic and topic-document. The first matrix describes the probability of selecting a particular word when sampling a particular topic. The second matrix describes the probability of selecting a particular topic when sampling a particular document. Therefore, the generative process builds the two matrices by estimating α and β.

3.1. Application to system log messages

In this work, LDA is applied to the messages extracted from the system log events of the System Under Test (SUT) and abstracted to remove contextual attribute values. A document is

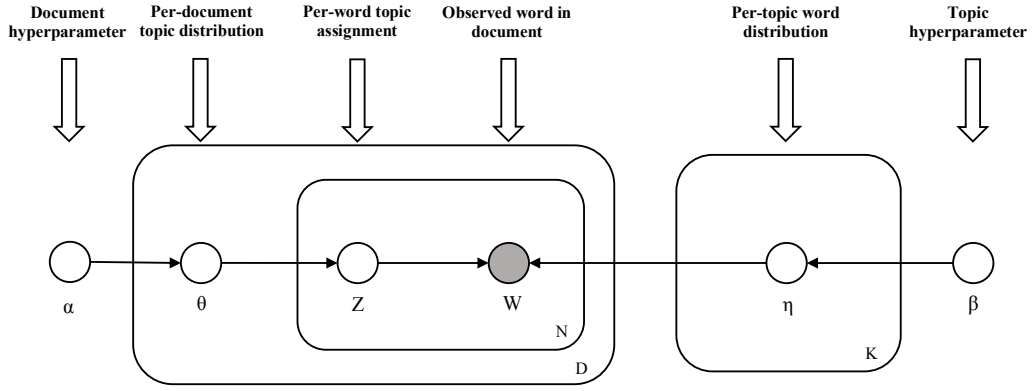

Figure 1: LDA model. Nodes represent random variables, observed variables are shaded, arrows represent possible dependence, plates are replicated structures. K= number of topics, D = number of documents, N = number of words

set of abstract messages of a sequence of logs corresponding to a SB. Words are terms contained in all documents and they form the vocabulary V. By running LDA, documents can be reconstructed each as a set of topics representing the thematic structure of documents. As such, each scheduling block SB, which represents one instance of the telescope's observation process, can be reconstructed as a set of telescope's tasks.

4. Study design

Our work uses an experimental protocol for exploratory analysis (Wohlin et al., 2012) on how to apply topic modeling to system's log mining. To this aim, in this section, we illustrate the protocol as 1) the research questions derived from the goal in Section 1.1, 2) the study context and the decision taken thereafter that drove the experiment settings and the analysis of the data, 3) the data collection and selection and description of the study samples, 4) the results are qualitatively described in Section 6 and are mainly demonstrative of the applicability and automation and 5) the approach for mining logs is finally described in details in Section 5.

4.1. Research questions

To achieve our research goal (Section 1.1), we formulate three research questions that we investigate by applying our approach to the ALMA system (i.e, the SUT) and its logs:

RQ1. Can the behavior of a system be reconstructed from system logs? We will model the behavior of a software system as sets of system's tasks obtained by mining the information carried by sequences of log events with LDA. We will additionally discuss what of our modeling approach can be fully automated.

RQ2. Can the proposed model of system behavior be used to associate code components to specific tasks performed by a system? We will able to localize software files and methods involved in the system's tasks identified in RQ1. Such knowledge will help software engineers to maintain the code of a system in relation to its behavior.

RQ3. Can the proposed model of system behavior be used to describe the cause of system anomalies? We show how to identify tasks as per RQ1 and code as per RQ2 responsible of system anomalies. In this work, anomalies are error events in logs and their causes are traced from the set of system's tasks of the proposed behavior models that contain the error. Providing such information to software testers will help them to design strategies to increase fault tolerance and predict future malfunctions of the system.

4.2. The Study Context

There is an ongoing interest of the ALMA team to automatically exploit the information available in system logs and better understand the complexity of the telescope's processes (Gil et al., 2016). We spent more than one year to analyze the context and the environment that required a certain knowledge of applied physics. In this section, we summaries the results of such analysis by describing the operational environment of ALMA and how context analysis has guided the selection of the samples for the study. We report in italic the major decisions derived from context analysis.

The ALMA Observation Process. The telescope's operations are triggered by the activities defined in the ALMA Observation Process. The process, the system and its sub-systems operating the telescope are described in Fig. 2. The process comprises all the phases and activities needed to scan and observe the universe. Astronomers around the world submit a project proposal for observing a specific astronomic event. Approved projects are executed during one cycle, i.e., a period of 12 months during which a set of technical capabilities of the telescope are guaranteed.

The measurements for a project are collected in one or more *Scheduling Blocks* (SBs). The observation process starts by manually selecting an array of antennas. An SB is selected from a list provided by the Scheduler sub-system and the execution starts. Then, the Control sub-system executes all the scripts of the SB and the scan / subscan sequences to command all relevant hardware. Each SB lasts from few minutes to 2

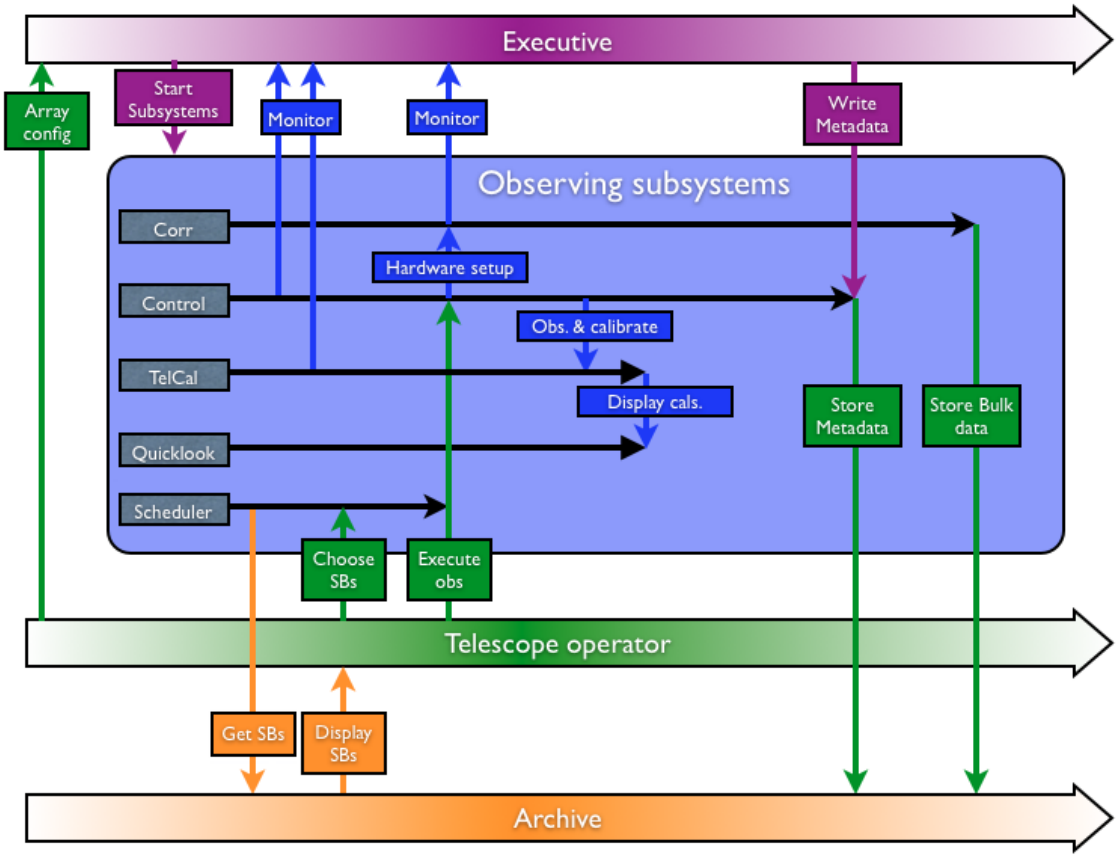

Figure 2: The Observation Process and the ACS architecture. Source: (Asayama et al., 2016)

hours maximum (on average 1.5 hours) and specific regions of the sky are scanned several times at the same frequency with the same set of antennas. The scripts use the information in the SB to determine the execution sequence as a set of scans and sub-scans of the universe. Each scan is compounded of several sub-scans lasting from few seconds to few minutes. On average, in each scan 10% of the time is dedicated to the real observation and 90% of the time to the calibration of the antennas. Each sub-scan specifies tuning, phase centering and antenna pointing, calibration device position, and intent metadata to convey the purpose of the scan/sub-scan. Thus, the data of an SB contains hardware calibrations, meta-data, and binaries obtained from the instruments. By internal policy, an SB runs to completion (i.e., it ends with an "end event"), fails (i.e., it ends with an "error event") or is interrupted by the Astronomer on Duty (and a new SB starts) (Asayama et al., 2017). The Control sub-system executes the scan / sub-scan sequences by commanding all relevant hardware. The resulting raw data and metadata are finally stored to the Archive sub-system and made available to the online Telescope Calibration (TelCal) sub-system that publishes, reduces, and archive the observation data. The QuickLook GUI sub-system displays the results to the operators and the Astronomer on Duty (AoD) to evaluate the observation progress. The Control sub-system stores the result of the SB in a compact form called Execution Block, which contains calibrations, meta data and binaries obtained from the instruments. When SBs are selected to start an operation process their Execution Blocks are retrieved.

An SB of the Control subsystem represents a complete behavior of the telescope and, thus, is the object of this study.

Science and engineering time. One of the major issues is identifying the time window from which to select the study samples. As mentioned in the challenges, such window must capture telescope's regular behavior (e.g., avoiding specific maintenance activities). The telescope operates in two modes: *science and engineering time*. Science time is dedicated to observations whereas engineering time is used for antennas' maintenance, software testing and general troubleshooting. Thus, we focused on science time.

Log events triggered during science time are considered in the study as they relate to the ordinary operations of the telescope.

The main users operating the telescope during science time are *operators*, AoD and *support astronomers* that form a mixed team between four to six persons. The team sets up software and hardware, performs initial calibrations, monitors the health of the system, acts as a first line of troubleshooting, and reports to software engineers all problems that the team cannot fix by itself. Thus, we selected log events trigged by such users.

Log events triggered by Operators, AoD and Support Astronomers are selected for this study.

The ACS architecture. ALMA is a distributed system consisting of six sub-systems, Fig. 2. In this study, we will focus on

the sub-system *Control* as it generates logs about the execution of SBs. The ALMA Common Software (ACS) is a middleware over the operating system. It implements the ALMA Container-Component model to support distributed development and runtime deployment of components, supplying services such as messaging, logging, error and alarm handling, and configuration database, (ALMA, 2017; Sommer et al., 2004).

The components run in the so-called containers. Containers run components that are logically connected and hosted in specific computers. There are containers for each development language used in ACS. For example, "CONTROL/ACC/javaContainer" refers to the sub-system Control, contains the code written in Java ("javaContainer"), and runs in the Array Control Computer (ACC), which is the computer located at the central control area at ATACAMA and responsible for coordinating all instrument activities. When a container is started, it instantiates a CORBA ORB and register itself to the ORB's Portable Object Adapter. The Manager Container provides the container's object reference for future requests and manage the requests to the components as a CORBA service by instructing a container to insatiate and run the requested component that it contains.

Typical software errors. A single telescope's observation is a complex process involving several hardware pieces and software applications that could be subject of failures at any instant. By the nature of the ALMA operations, the majority of the errors refer to a timeout of a software / hardware component. Examples of timeouts are:

- more than 48 ms for monitor and control hardware devices as part of Controller Area Network (CAN bus) (e.g., antennas, inner devices, Correlator sub-system, etc.)
- More than 120 seconds for sub-scan duration
- More than 5 minutes for scan duration
- More than 5 minutes for CORBA timeouts, that is, when communicating over ACS fails for any reason

Such timeouts do not always trigger an error that is submitted to the ticketing system, but they may be traced in the logs of the ALMA system.
In this study, we focus on errors that can be traced in logs.
Logs. In ACS, logs are coded in XML files where each log event is a node whose attributes consist of several structured data and an unstructured XML character attribute `CDATA` that contains the log event message, Listing 1. An XML node tag describes the types of log. Among such types, we focused only on the ones targeted to operators, AoD, or Support Astronomers (e.g., "INFO") or tagging errors (e.g., "ERROR", "WARNING" etc.). The source code information is stored in the node attributes "File", "Line", and "Routine". The runtime context information is stored in the fields "Host", the name of the computer from which the log entry is triggered, "Process", a container, "Thread", a thread in a component, and "SourceObject" an ACS component. The log entry message is a string of characters enclosed in the `CDATA` field.

Listing 1: An example of log event text produced by the ACM system

```
<INFO
    TimeStamp= 20170605T05:54:05.520Z
    File= ObservingModeBaseImpl.java
    Line= 260
    Routine= beginSubscan
    Host= gas01
    Process= CONTROL/ACC/javaContainer
    SourceObject= CONTROL/Array001
    Thread= Thread771
    LogId= 20375
    Audience= Operator >
<![CDATA['Scan 5, subscan 3 has an intent of HOT, takes
5.760 seconds from 05:54:05.520 to 05:54:11.280']]>
</INFO>
```

This study focuses on the types of log events tagged as "INFO", "ERROR", and "WARNING".
Manual mining logs to search cause of error. Fig. 3 illustrates the complex process used by ALMA engineers to query their three databases (called JIRA, ICTJIRA, Kibana) and isolate those log events that may explain the cause of an error. JIRA database is based on JIRA software[2] and contains tickets that the operators submit when they observe some system misbehavior. Tickets are labelled with the identifier PRTSPR-ID (e.g., PRTSPR-9280). ICTJIRA is the database of tickets that software engineers and developers already investigated. These tickets are labelled with the identifier ICT-ID (e.g., ICT-3210) and include a very detailed report on the cause and solution of the issue. Log events are collected and stored with Logstash[3] in the Kibana database, searched with the engine Elasticsearch[3], and visualized with the Kibana tool[3]. The adopted configuration of the Kibana tool only visualizes time series of log events.

Either in science time and engineering time, many problems cannot be solved directly by operators. For such cases, there is a support team of specialists including software engineers that intervenes when a new ticket is submitted to the ALMA ticketing system as described in Fig. 3. The first goal of software engineers is to check whether similar tickets have been reported in the database and if the new ticket can be inspected by mining logs (top box in Fig. 3) then they search the cause of the issue by manually inspecting logs (bottom box in Fig. 3). The process starts by choosing a new ticket in the JIRA database (say PRTSPR-120) Then, the ICTJIRA repository is queried by title or title fragment of the selected ticket to find previous solved tickets on the topic. If any, the most recent ticket from ICTJIRA (say ICT-203) and its associated (closed) ticket form JIRA (say PRTSPR-100) are selected. then, new tickets including PRTSPR-120 and similar to PRTSPR-100 (say PRTSPR-115,...,PRTSPR-125) are selected from the JIRA database. By reading the information in all such tickets, software engineers can decide on their own experience whether these new tickets can be analyzed by inspecting logs. Not all tickets can be investigated with software logs. There is a broad class of problems (about one third), like the ones related with energy supply, host down, network issues, that are investigated in other ways. These classes of problems are not matter of this study. If a ticket can

[2]https://www.atlassian.com/
[3]https://www.elastic.co/products/

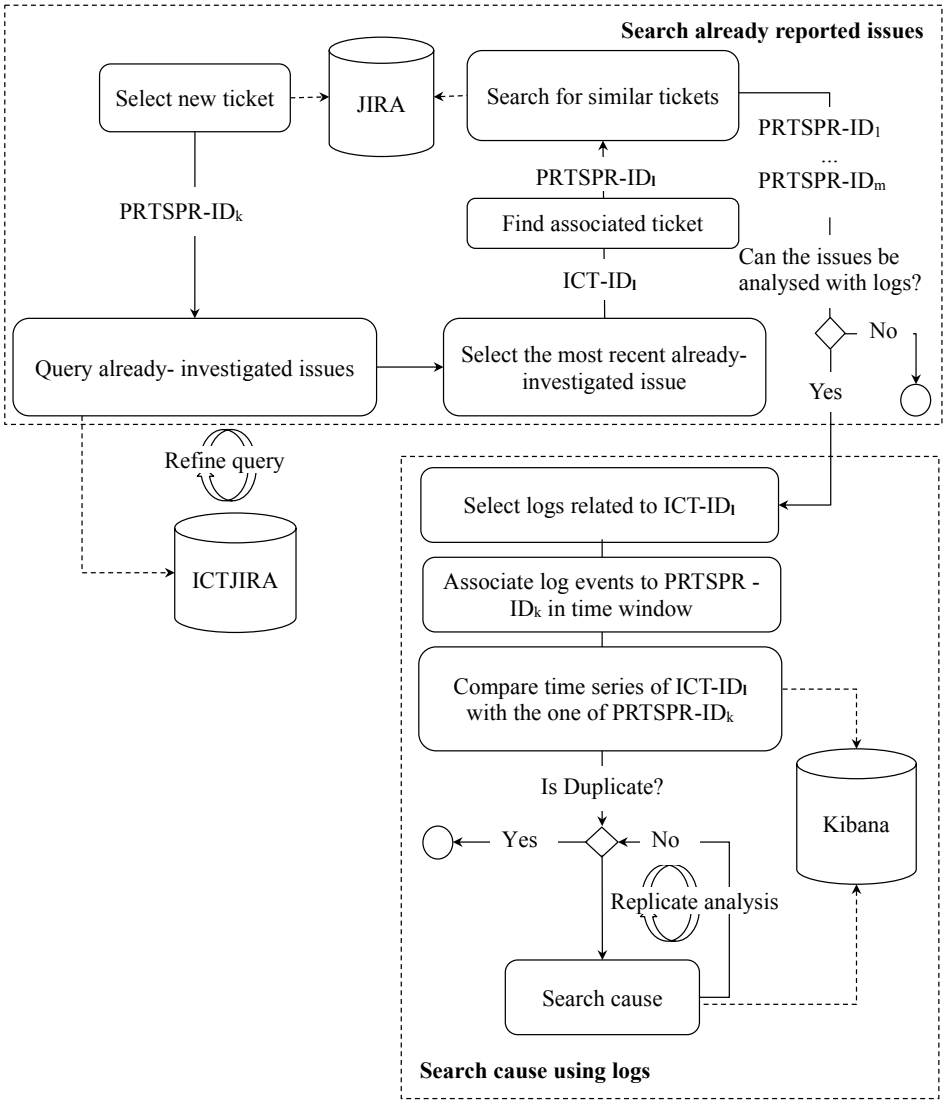

Figure 3: The existing manual process at ALMA to isolate the cause of an error by exploiting system logs.

be investigated with logs, the process to search the cause of an error (bottom box in Fig. 3) is started by select the logs relevant for the solved ticket ICT-203. Time windows in which to examine the log events are then chosen by inspecting the information contained in the new tickets, PRTSPR-115,...,PRTSPR-125. By visual inspection with Kibana tool, the support team compares the time series of events associated to the solved issue ICT-203 with the one associated to any of the new tickets PRTSPR-115,...,PRTSPR-125 and decide whether any of the new tickets is a duplicate of the closed one PRTSPR-100. If this is not the case, log events are again visually inspected with the Kibana tool. Anytime the frequency of a log event in such plots is unordinary high in the opinion of software engineers, the ticket is tagged as bug, then solved, and a new report is submitted to ICTJIRA.

Therefore, the process to identify the tasks of the telescope that are relevant to solve the ticket may be biased by the human perception and subjectivity.

4.3. Study Samples

We selected two samples of logs: a testbed sample and a validation sample collected in two different days of the same week. The testbed sample is used to illustrate the proposed method, whereas the validation sample is used to ensure that the vocabulary for topic modeling obtained from the testbed sample is not related to specific states or operations of the telescope (Section 8).

The **testbed sample** contains log events triggered on 2017.03.13. Such events occurred around the time of occurrence of a bug issued during the normal operations of the telescope and stored in ICTJIRA. The bug is about the too long initialization of meta-data model that must be later saved in the database of astronomical data. It was selected because the bug occurred a few times between the open and fix date of the bug report. Thus, software engineers could observe logs while the bug re-occurred and include in the bug report the information on the log events related to such bug occurrences. From the

Table 1: Description of the samples.

	Testbed sample	Validation sample
Bug report ID	ICT9453	NE
Time window	20170312T23:33:47.219 - 20170313T04:53:55.866	20170313T23:00:11.943 - 20170314T22:54:19.649
Log size	174 MB	1,14 GB
Total no. events	1,752	18,214
Log priority(count)	ERROR(295), INFO(1,386), WARNING(71)	EMERGENCY(2), ERROR(2,543), INFO(15,239), WARNING(430)
ARRAYS	001,005,007,008	009, 011,016, 018,026

information carried in such logs, we first infer the sub-systems and components relevant for the analysis. We select the "CONTROL/ACC/javaContainer" (see Section 4.2) that includes the Java code that orchestrates all the instruments' activities of different arrays of antennas. Each array (e.g., A007) includes antennas that perform the same data collection, have the same frequency, point the same part of the sky, and use the same hardware (e.g., sub-reflectors). Thus, with the XML log parser we wrote (Section 5.11), log events in the testbed sample were selected by the node attribute *Process* equal to "CONTROL/ACC/javaContainer," the node attribute *SourceObject* that contains the term "ARRAY" and the node type (i.e., Log type) not DEBUG (i.e., logs related to ordinary operations of the telescope). In total, log events within about 5 hours, 16 complete SBs were collected and seven arrays of antennas. The **validation sample** is collected one day after and for about 24 hours according to the same rules.

The two samples are described in Table 1 and Table 2.

5. Study Approach

The approach proposed as the contribution of this paper is overviewed in Fig. 4 and discussed in the following sections.

The approach is designed to be used for any system whose log events carry similar information and structure (e.g., code data) as for our SUT and the system behavior can be inferred from events' sequences (e.g., with begin and end events). Domain experts have also a key role in the proposed approach as the domain is highly technical and the validation of the model requires insider's knowledge.

The approach also aims at automatizing the most the manual process currently performed by the software engineers at ALMA and described in Fig. 3. The output of this approach are behavior models constructed as set of telescope's tasks (latent topics) and code components involved in such tasks.

5.1. Approach overview

The proposed approach (Fig. 4) starts by selecting a sample of log events. As illustrative example, we have selected a sample of log events within a time window containing the occurrence time of a bug reported in the issue tracker JIRA of the ACS system (as described in Section 4.3). In this way, log events related to at least one anomaly (i.e., the reported bug) are included in the chosen sample. We first identify the sequence of log events related to each SB as described in Section 5.4. For these events, we extracted the event messages from the XML nodes and abstracted them as described in Section 3. The abstracted messages of each SB are merged to create an input document for LDA analysis, as portrayed in Section 5.4. With LDA analysis, topics are identified and a map between individual event messages in a document and topics is defined through the maximum LDA posterior probability (see Section 5.6). Thanks to this map, an appropriate number of topics is determined (Section 5.6) and patterns of sequences of messages recurring over SBs are retrieved to label topics, Section 5.8. Methods and files are then associated to topics by linking log messages to topics when they have positive posterior probability, Section 5.9. Through the same association, the number of the messages associated to a topic is used to reconstruct a document as set of topics (see Section) and define the behavior models. Finally, the analysis of models and their associated code components are used to describe the telescope's behavior and localize anomalies (Section 6).

5.2. Sequences of log events corresponding to SBs

Identifying sequences of log events corresponding to specific SUT behavior is always a challenge (Russo et al., 2015). On the one hand, log events often do not explicitly mention the start and end of a SUT behavior and events of different SUT executions may interleave making hard the identification of a specific SUT behavior. On the other hand, a SUT anomaly may originate and propagate in events preceding the one in which it manifests itself. Thus, one typical problem in log mining is to split logs in sequence of events each determining a specific SUT behavior. In this study, after we interviewed the ALMA software engineer and explored the logs, we were able to split logs by sequences of time-ordered events corresponding to scheduling block of the Observation process (Section 4.2). We called such sequences, *SBs*, as the corresponding scheduling block. To automatically retrieve SBs, we first filter logs as described at the end of Section 4.2 to avoid selecting events of different SUT executions and then we split the logs into sequences of events that occur between two log events with message `Beginning SB execution.` According the ALMA software engineer, the resulting SBs correctly describe individual scheduling blocks of the Observation process.

5.3. Message abstraction

To recognize patterns in sequences of messages, we abstract messages to remove contextual attribute values, leaving data flow information that captures the rationale underlying the use of these values. This is a typical technique of text preprocessing, (Russo et al., 2015; Jiang et al., 2008; Mariani and

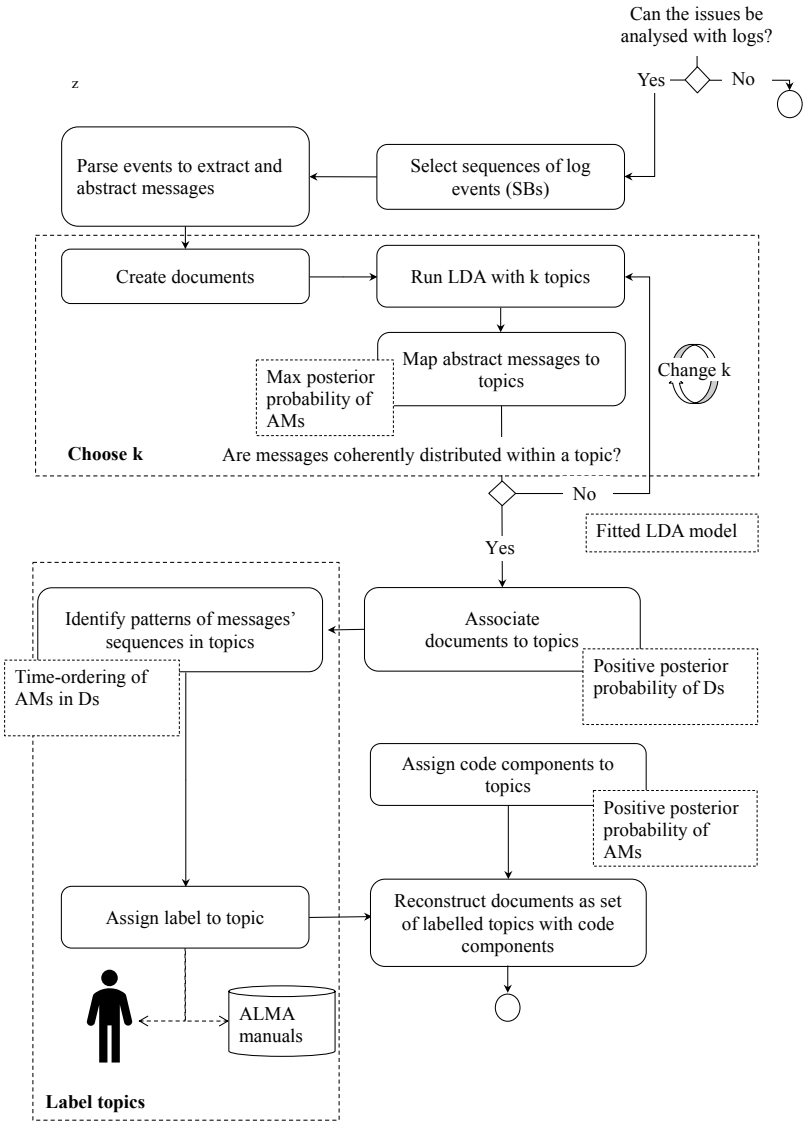

Figure 4: Approach overview to reconstruct SUT behavior by mining log events with LDA.

Pastore, 2008). In this work, event messages are abstracted by keeping the static information that describes the event (e.g., "Start scan") and removing the dynamic values generated at run-time (e.g., the scan progressive index). The result of this process is illustrated in Listing 3, the original message, and Listing 2, its abstraction.

Listing 2: Log message in the CDATA field.

```
Tuning BB_1 to 221.538GHz BB_2 to 223.538GHz BB_3 to
237.538GHz BB_4 to 239.538GHz (Band 6) & optimize the IF
power levels to 30.0 dBm and the BB power levels to 2.4 dBm
 and label this setting Band 6 pointing/focus at 03:47:17
.322 for a correlator calibration subscan
```

Listing 3: Abstraction of log message in Listing 2

```
Tune [BBs], Optimize [IF power level, BB power level],
Label [settings Band Device - Pointing/Focus]
for [Correlator Calibration]
```

where values in '[ ]' are predefined constants described in the reference documentation (Asayama et al., 2017). Such constants qualify the action performed by the telescope. For example, [BBs] stands for Base Bands of signal frequencies and [settings Band Device - Pointing/Focus] is the setting of a device for one of the calibration types (i.e., Pointing/Focus). By removing contextual references, abstraction allows to compare messages and find the same message or recognize patterns within sequences of event messages and across different SBs. After abstract abstraction the vocabulary of corpus consists of 114 distinct words in the testbed and 169 in the evaluation sample.

5.4. Documents for LDA analysis

The context and the goals of a research typically determine the granularity of the input (i.e., Documents) for LDA analysis, (Mehrotra et al., 2013). In our study, we excluded individual

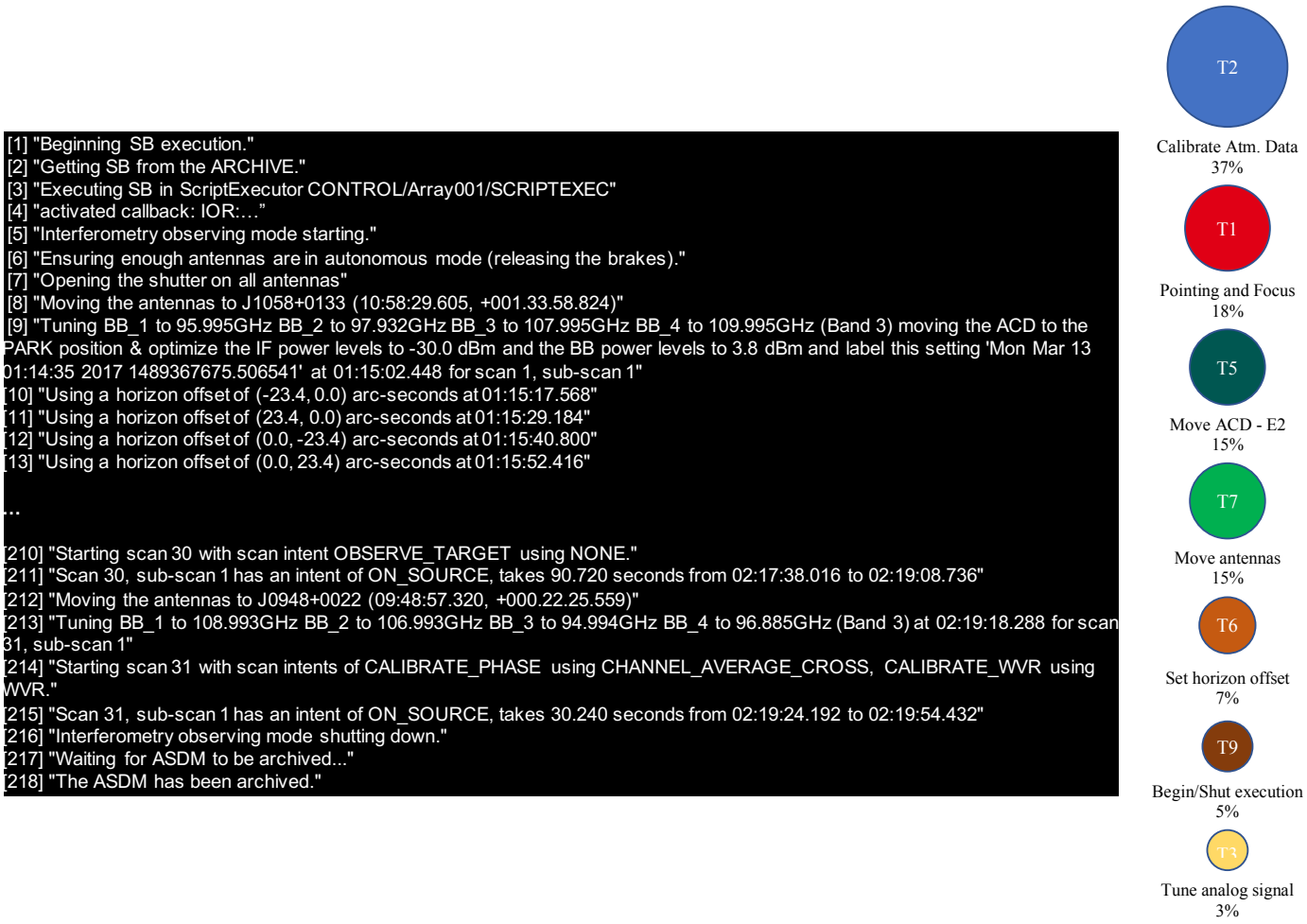

Figure 5: The effects of topic modeling. On the left, the 218 time-ordered log messages describing the execution of SB_4; on the right, the seven labelled topics characterizing SB_4 ordered by the percentage of messages in SB_4 mapped to each topic.

event messages as input since short texts (like event messages or tweets) do not typically carry enough vocabulary to apply topic modeling with success. Fig. 5 on the left shows the consecutive messages for SB_4. By reading the messages in the figure, we can see that different types of events are repeated and intermixed. In addition, some messages as read individually have a rather obscure meaning (e.g., "Scan 30, sub-scan 1 has an intent of ON_SOURCE, takes 90.720 seconds from 02:17:38.016 to 02:19:08.736"). As mentioned in Section 3, recent literature opted for strategies based on clustering of short messages. In this research, we adopt the *temporal pooling schema for text processing improvement*, which is a strategy to aggregate event messages according to some temporal order (Zou and Song, 2016). Specifically, we defined a document by merging abstract messages of consecutive events of an SB.

In this work, a document is obtained by merging abstract messages that correspond to an SB. Specifically, *a document D_j is a set of abstract messages AM_i of events e_i, which occur between two log events with message* `Beginning SB execution`. Table 2 describes the sequences selected for this study. To build a document, log messages are extracted by parsing log events $\{e_i\}$ of an SB_j with a regular expression that searches the text encapsulated in the `CDATA` field. Then each event text is abstracted as described in Section 3. An LDA document D_j is finally built by merging the abstract messages $\{AM_i\}$ of the SB_j:

$$SB_j = \{e_1, ..., e_s\}, \; D_j = \{AM_1 \cup ... \cup AM_s\}$$

Fig. 5 on the left shows the document for SB_4.

Table 2: Description of the documents

	Testbed sample	Validation sample
# events in SBs	1,485	13,462
# Documents	16	64
Document's length (min, max, avg)	1, 218, 95	5, 626, 221

5.5. LDA hyper-parameters α and β

We use one of the Grün and Hornik methods (Gruen and Hornik, 2011) to estimate the parameters of the Dirichlet distributions while we fix a priori the value of k. Specifically, we

use the Variational Expectation Maximization (VEM) with values for the parameters $\alpha = 50/k$ and $\beta = 0.1$ as recommended in (Griffiths and Steyvers, 2004b). The Grün and Hornik algorithm performs the following pre-processing on the input corpus prior to constructing the document-term matrix:

- Converting all terms to lowercase
- Removing punctuation and numbers
- Removing stop words using the SMART list (Benoit et al., 2017)
- Porter stemming terms using the Snowball algorithm (Porter et al., 2019)
- Omitting terms with length < 3 as suggested in (Menzies and Marcus, 2008).

5.6. Message - Topic mapping

The Message - Topic mapping aims at representing topics as sets of abstract messages. Firstly, an LDA model with k topics is trained on the documents of the testbed sample. A set of k topics T_1, ..T_k is therefore determined. Then, iteratively over all abstract messages, if a message contains an as-yet-uncovered vocabulary, its LDA posterior probability is computed (Blei et al., 2003). Finally, an abstract message is univocally associated to a topic, if the message has its *maximum posterior probability* for that topic. In this way, a topic T_i is represented by a set of unique abstract messages with their maximum posterior probability for T_i:

$$T_i = \{AM_{a_1}, ..., AM_{a_r}\} \tag{1}$$

As messages are mapped to topics by their highest posterior probability, there might be one or more topics for which no messages are mapped to and such topics are mapped to the empty set. In addition, by construction and empirical observation, two topics have non-overlapping sets of abstract messages (i.e., no messages happen to have the same maximum probability on two different topics). Then, abstract messages mapped to a topic are clustered by documents they belong to. Thus, a topic T_i is represented as a set of sets of abstract messages:

$$\begin{aligned} T_i &= \{D_1, ...D_n\} \\ D_j &= \{AM_{v(a_1)}, ..., AM_{v(a_r)}\} \end{aligned} \tag{2}$$

Each document corresponds to an SB in which messages are ordered by the time stamp of the events they belong to. We used this information to identify sequences of events in SBs that corresponds to messages in each document of a topic as represented in 2. Table 3 shows the result of this process for topic T_5 in which all message abstractions are assigned a letter of the alphabet. Table 3 shows the abstract messages (A-I, W, E2) mapped into topic T_5 and the sequences of such messages for each SB in which the messages appear ordered by their time stamp (i.e., SB2, SB4, SB15, SB16).

Table 3: Message patterns in Topic 5: "Move ACD."

```
Messages in Topic 5:
(A)  Tune [BB]
(B)  Move [ACD] to [AMBIENT LOAD] for [scan,
     subscan]
(C)  Load [Attenuator settings] [Band device delay
     Load Attenuations] for [scan, subscan]
(D)  Move [ACD] to [HOT LOAD] for [scan, subscan]
(E)  Load  [Attenuator settings] [Maximum
     Attenuator Setting] for [scan, subscan]
(F)  Move[ACD] to [PARK] for [scan, subscan]
(G)  Optimise, label [IF, BB power level]  for [
     scan, subscan]
(H)  Start Scan [CALIBRATE-POINTING, CALIBRATE-
     WATER VAPOR RATIO] use [CHANNEL-AVERAGE-CROSS,
     WATER VAPOR RATIO]
(I)  Start Scan [CALIBRATE PHASE, CALIBRATE WATER
     VAPOR RATIO] use [CHANNEL-AVERAGE-AUTO, WATER
     VAPOR RATIO]
(W)  Timed out ACD to get into position.  Not
     shutdown the observation.  Data will be
     flagged & subsequent power level optimizations
     may be incorrect
(E2) Operation Timeout (type=10000, code=14)

Sequences of event messages by document in Topic 5:
D2:
HABCDE E2W
D4:
AFCG H BD E2W BD E2W BD E2W IBD E2W IID E2W IBD E2W II
D15:
AFCG H BD E2W BD E2W BD I BD E2W II BD E2W I BD E2W II
D16:
AFCG H BD E2W BD I E2W  BD E2W I BD E2W II BD
```

5.7. The number of topics k

Finding the number of topics k is an open issues (Layman et al., 2016; Agrawal et al., 2018; O'Callaghan et al., 2015; Koltcov et al., 2014) as topics must be semantically distinct as well as cohesive (O'Callaghan et al., 2015; Blei et al., 2003). In literature, approaches to evaluate topic coherence and independence vary from direct approach by asking people about topics to indirect approach by evaluating point-wise mutual information between the topic words (O'Callaghan et al., 2015; Lau et al., 2014), and automated approach by using measures like perplexity (Chang et al., 2009) or searched based software engineering such as differential evolution Agrawal et al. (2018) or simply by examining the results for a few different numbers of topics, Damevski et al. (2018). For example, (Chang et al., 2009) found that perplexity was often negatively correlated with human judgements of topic quality and suggested that evaluation should model human intuition. Thus, the process of choosing the number of topics still lacks definitive guidance, (Layman et al., 2016; O'Callaghan et al., 2015; Zhao et al., 2015). In this work, we propose a method to choose k by assessing the cohesiveness and independence of topics once they have been

Table 4: Top-5 words of the nine topics obtained with LDA on the testbed.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
1	refer	block	stop	stop	request	schedul	request	stop	request
2	maximum	schedul	schedul	schedul	stop	receiv	stop	receiv	stop
3	channel-average-auto	request	request	receiv	block	stop	schedul	schedul	receiv
4	calibrate-ampli	receiv	block	block	maximum	block	receiv	block	schedul
5	read	stop	receiv	request	receiv	request	block	request	block

represented as a set of abstract messages by the Message -Topic map. Reading topics as set of messages instead of individual words helps understanding their semantic and verifying their coherency and independency. Thus, our method iteratively

1. Chooses the number of topics k, and the non- negative parameter α and β as in Section 5.5,
2. Runs LDA on the documents for k topics
3. Represents topics as set of abstract messages by the Message -Topic map
4. Assesses topics for cohesiveness and independence,
5. Re-runs the LDA algorithm with different k value if the result of step 4) is not satisfactory.

Each set of messages representing a topic is evaluated for cohesiveness and independence according to three assumptions:

A1 Messages that carry information of the same error must be in the same set (i.e the same topic);

A2 Information carried by messages in the same set must be coherent;

A3 Messages in different topics must be semantically uncorrelated.

As in literature (Griffiths and Steyvers, 2004a), we start with $k = 100$. With $k = 100$, only 22 topics were found relevant

(i.e., non-empty set under the Message-Topic map) and a further manual inspection revealed low cohesiveness of such topics. For instance, all log messages related to the task "synchronize antennas" were scattered in four different topics (violating A3). Thus, we selected a value of the parameter k lower than 22. When we reached $k = 10$, the assumption A2 and A3 were first satisfied except in the case of messages that initialized an SB were erroneously distributed over two different topics (violating A3). This did not happen for $k = 9$. In addition, the two distinct messages "Error reading input data (type=30001, code=8)" and "Operation Timeout (type=10000, code=14)" referring to two different errors were also correctly mapped into two distinct topics (according to A1). One topic (T_4) was empty though. Therefore, we built the Message-Topic map for $k = 8$. In this case, few non-related messages were moved into the same topic violating A2. Therefore, in the end, we selected $\mathbf{k = 9}$.

5.8. Topic labelling

Labelling a topic means associating to a topic a label expressing its semantic (Lau et al., 2011). The typical approach for labelling topics is analyzing the distribution of words in each topic with the help of domain experts (Layman et al., 2016; Hindle et al., 20). In the case of the ALMA logs, this procedure proved not to be successful as the differences among the terms of the topics were too small to uniquely identify a topic by visual inspection. This can be easily seen from the top five words in the nine topics obtained by LDA on the testbed sample in Table 4. Thus, the approach we propose exploits the Message - Topic mapping to enrich the information contained in each topic and, in this way, facilitate the label assignment. The approach is based on the recognition of patterns of messages in topics. As a topic can be represented as sets of abstract messages (Eq. 2) that are ordered by the time stamps of their events, patterns can be found as sequences of abstract messages that recur over such sets. Thus, a topic can be described by its patterns. As patterns are more informative than bags of words, they can provide, in principle, more accurate topics' labels.

Table 3 illustrate the output of our method for T_5. On the top of the table the messages with maximum posterior probability are shown. On the bottom, sequences of such messages found over the documents of the corpus are illustrated. Documents that do not include any of such messages are not listed.

T_5 has two sequence patterns *AFCG H* and *BD E2W*. Such patterns occurs with little variation in almost all SBs of T_5 and never in the other topics (omissis). As such, the two patterns characterize the topic and can be used to label it. According to the ALMA manuals and the ALMA software engineer, T_5 was given the label "Move ACD" as AFCG H and BD refer to the typical operation of the Alma Calibration Device (ACD) to capture atmospheric weather conditions: after the device has been calibrated (corresponding to the pattern AFCG H), a mechanical arm replaces an ambient microwave absorber (AMBIENT LOAD, 20C) with hot microwave absorber (HOT LOAD, 70 C) in front of the receiver feed horns of any of the signal band devices (Base Bands), (corresponding to pattern BD) (ALMA-Newsletter, 2012). Then a warning and an error appear (corresponding to the pattern E2W). Thus, the topic T_5 also contains an error. The same procedure has been performed for all topics and their labels are illustrated in Table 5. As topics must be independent (Section 5.6) and the value of k has been chosen to respect such principle, the fact that we found different patterns across the topics (omitted for space reasons) further validates the choice we made for the value of k.

Table 5: Topics' labels.

Task	Label
T_1	Pointing and Focus
T_2	Calibrate Atmosphere data
T_3	Tune Analog Signal
T_5	Move ACD
T_6	Set Horizon Offset
T_7	Move Antennas
T_8	Error
T_9	Begin/Shut Execution SB

5.9. Associating files and methods to topics

To associate code components to topics, abstract messages are again mapped to topics through their posterior probability. All messages with posterior probability greater than zero for a topic are analyzed and all code components valued in fields "file" (i.e., code file) and "routine" (i.e., code method) of the events of such messages are associated to the topic as in the following:

$$T_i = \{f_{h_1}, m^1_{h_1}, ..m^d_{h_1}, ..., f_{h_t}, m^1_{h_t}, ..., m^g_{h_t}\}$$

where f_j is a class and m^k_j is a method of this class. Unique files and methods among the ones in a topic are finally associated to that topic. Worth noticing here that the same files or methods can be associated to more topics. The final association is illustrated in Table 6.

5.10. Reconstructing an SB as a set of tasks

LDA model outputs a Document-Topic matrix that describes the probability to find a topic in a Document, Section 3. Through this matrix, a Document can be reconstructed as a set of topics with non-zero probability; correspondingly an SB is reconstructed as a set of tasks of the telescope. Table 7 illustrates the SBs as set of tasks for the testbed sample.

For each SB in the table, the percentages are calculated as the number of abstract messages AM of the corresponding LDA document D that have non-zero posterior probability for a topic T_i ($prob^i > 0$) over all messages of D:

$$rel^i = \frac{|AM \in D \ \& \ prob^i(AM) > 0|}{|AM \in D|}. \quad (3)$$

In other words, the metric *rel* quantifies the probability of an SB to include a task in terms on the number of events associated to the task.

Table 7 groups SBs by their tasks. Each of the five groups represents a *behavior model* M_i of the telescope as it corresponds to the same set of tasks. The table shows that models do not depend on the hardware setting (array of antennas) or time window (index of the SBs). Three error are found: E1, E2, E3. Each in the same tasks and in different models (M_2, M_3, and M_4). The error retrieved in the JIRA database (E3) is not reported as an error event, but it stops the execution of the SBs of model M_4 (i.e., only T_9 "Begin/Shut Execution SB" is included in the model).

The output of our approach is a behavior model that consists of a set of tasks of the telescope independent from specific hardware setting or observation time. Classes and method are univocally associated to each tasks.

5.11. Tools for mining logs

We implemented few tools to perform our empirical study and mine logs. A Java program parses XML files. To prepare the data and run LDA, two R packages *topicmodels* and *quanteda* are used. A final R script maps individual messages to topics and draw the distribution of topics in an SB[4].

6. Results

In this section, we illustrate the application of the method to the 1752 log events of the testbed sample, Table 1 and answer the research questions.

RQ1. Can the behavior of a system be reconstructed from system logs? With the procedure described in Section 5.10, we reconstruct an SB as a set of tasks performed by the ACS system, where an SB corresponds to a document D and tasks correspond to the topics T_i found by the LDA model. The *behavior model* is then defined by a specific set of tasks recurring over SBs. Table 7 illustrates the behavior models of the ACS system derived from the SBs of the testbed sample. The SBs are grouped by types of tasks that compound them. There are two groups of SBs with no error and three with error. Each of these groups is a behavior model. Worth noticing here that T_9 represents the task that always opens and closes the execution of any SB (Fig. 5) and, thus, it is not crucial to characterize the semantic of a specific SB. The SBs in the first group tune the analog signal (T_3) and move the antennas (T_7) before pointing and focusing them (T_1). In these tasks, no error is reported, whereas additionally setting the horizon offset (T_6) causes an error (E1) in group three. If, further, the ACD is moved (T_5) to calibrate the atmosphere data (T_2), another error (E2) occurs in group two. The error we found in the ticketing system (E3), completely blocks the execution of the SBs (group four). Finally, only moving the antennas (T_7) is executed with no errors (group five).

RQ2. Can the proposed model of system behavior be used to associate code components to specific tasks performed by a system? With the procedure described in Section 5.9, we associate files and methods to tasks (Table 6) and, in turn, to the SBs that execute those tasks (Table 7). Files and methods that uniquely appear in a task are reported in boldface in Table 6. Other methods are, instead, shared by the topics as they perform a more generic action. For example, the method `ObservingModeBaseImpl.java::beginSubscan` typically initiates an operation of the telescope; thus, for example, it occurs to start setting the horizon offsets in T_6 as well as pointing and focusing the antennas in T_1.

[4]The code can be found here https://goo.gl/JCWUhE

Table 6: Code files and methods per topic. Methods in boldface indicate components solely associated to a topic.

no.	Label	Files and methods
T_1	Pointing and Focus	ObservingModeBaseImpl.java::beginSubscan, LocalOscillatorThread.java::generateTuningInfo, LocalOscillatorImpl.java::setAttenuators, **ArrayMountControllerImpl.java::setSubreflectorPositionOffset**
T_2	Calibrate Atmosphere Data	ObservingModeBaseImpl.java::logBeginScan,::beginSubscan, LocalOscillatorThread.java::generateTuningInfo, LocalOscillatorImpl.java::optimizeSignalLevels,::setAttenuators
T_3	Tune Analog Signal	LocalOscillatorThread.java::generateTuningInfo, LocalOscillatorImpl.java::optimizeSignalLevels
T_5	Move ACD	ObservingModeBaseImpl.java::logBeginScan, **LocalOscillatorThread.java**::generateTuningInfo,**::waitForACD**, LocalOscillatorImpl.java::setAttenuators, **AntennaCallbackBase.java::waitForEnoughResponses**
T_6	Set Horizon Offset	ObservingModeBaseImpl.java::logBeginScan,::beginSubscan, LocalOscillatorImpl.java::optimizeSignalLevels, **ArrayMountControllerImpl.java::setOffsetPrivate**
T_7	Move Antennas	ObservingModeBaseImpl.java::logBeginScan, LocalOscillatorThread.java::generateTuningInfo, **InterferometryObservingModeImpl.java::init**, **ArrayMountControllerImpl.java::setDirection**
T_8	Error	**ArrayStateBase.java::reportFocusModel**
T_9	Begin/Shut Execution SB	**InterferometryObservingModeImpl.java::init,::cleanUp**, **AutomaticArrayOperationalState.java::run**, **ArrayMountControllerImpl.java::track,::openShutter**

Table 7: Behavior models for the SBs in the testbed. SBs are indexed in temporal order (e.g., SB_1 precedes SB_2). Values in the topics' columns are percentages computed with Eq. 3.

Model	SB ID	Length	Array	T_1	T_2	T_3	T_5	T_6	T_7	T_8	T_9	Error
M_1	SB_1	38	A001	11		29			34		26	No
	SB_6	50	A003	8		48			24		20	No
	SB_7	50	A003	8		48			24		20	No
	SB_9	50	A005	8		48			24		20	No
	SB_{10}	38	A007	11		29			34		26	No
	SB_{11}	38	A007	11		29			34		26	No
M_2	SB_2	66	A001	24	6	9	**9**(E2)	20	17		15	E2
	SB_4	218	A001	18	37	3	**15**(E2)	7	15		5	E2
	SB_{15}	217	A007	18	37	3	**15**(E2)	7	15		4	E2
	SB_{16}	154	A007	17	34	5	**16**(E2)	10	14		5	E2
M_3	SB_3	213	A001	39		1		11	4	**39**(E1)	5	E1
	SB_8	150	A004	56		4		16	5	**12**(E1)	7	E1
	SB_{14}	217	A007	39		1		11	4	**41**(E1)	5	E1
M_4	SB_{12}	1	A007								**100(E3)**	E3
	SB_{13}	8	A007								**100(E3)**	E3
M_5	SB_5	12	A002						17		83	No

RQ3. Can the proposed model of system behavior be used to describe the cause of system anomalies? As we mentioned (Section 4.1), this study focuses on anomalies that can be mined from log. In the ALMA logs, such anomalies are identified by events that either stop the execution of an SB or are tagged as errors or warnings in the XML logs. In the former case, SBs do not complete and no shutting event is reported in the log (i.e., the last three messages in Fig. 5 do not occur). This is the case of the error reported in the ticketing system JIRA that we used to select the testbed sample (Section 4.3). The latter case, errors are completely identified by their type and code as described in the ALMA manual, ESO (2016). The value of the type is number that can be chosen between 0 and 909999 and among ranges that identify general ACS errors (0-9999) and ALMA subsystems' errors (e.g., for CONTROL is 10000 -19999), as illustrated in Table 8. The message further includes a pre-defined code (the list of codes is maintained in ALMA manuals, ESO (2016)) and a short free-text description. An example of error message is:

```
Error reading input data (type=30001, code=8)
```

Overall, the information carried by the error events is minimal and often requires good knowledge of a specific subsystem.

In Table 3, task T_5 is an anomaly. It contains the error E2 whose message is

```
Operation timeout (type=10000, code=14)
```

The type refers to the "CONTROL" subsystem, Table 8 and the pre-defined code is described in the manual as "Any TICS TCorr error", (ESO, 2016). The free text reports the timeout of

Table 8: Ranges of values for error types.

subsystem	range
ACS	0 - 9999
CONTROL	10000 - 19999
CORRELATOR	20000 - 29999
OFFLINE	30000 - 39999
TELCAL	40000 - 49999
PIPELINE	50000 - 59999
ARCHIVE	60000 - 69999
EXECUTIVE	70000 - 79999
SCHEDULING	80000 - 89999
OBSPREP	90000 - 99999
HLA	100000 - 109999
ACA	110000 - 119999
Examples/Test	900000 - 909999

a not specified operation. As such, this description is not much informative. We argue that the behavior models built with our approach (Table 7) and the associated code components (Table 6) can provide better insight about errors and their causes.

In the reminder of this section, we first overview the information delivered in Table 7 and then we discuss each error by comparing the model in which it has been found with the error-free model M_1.

Firstly, Table 7 shows that our approach is able to encapsulate each error in a single task of a single behavior model in a consistent manner independently from hardware configuration and observation time window. (e.g., error E2 occurs only for model M_2 and for any Sb of the model) The table shows that three types of errors, (E1, E2, E3), were found in the testbed sample. E3 is the error that was retrieved from the ALMA ticketing system when the testbed was sampled (see Section 4.3). E1 and E2 are two different error events found in testbed's logs. Each of the three errors consistently occurs in the same task of different SBs (e.g., E1 occurs in T_1 for SB_3, SB_8, SB_{14}) and in the same behavior model (e.g., E1 occurs in M_3). The three errors occur in different tasks, for different arrays of antennas (e.g., E1 occurs for the arrays A001, A004, A007), and during different timeframes also non-consecutive (the index of the SBs is ordered by time, e.g., SB_1 precede SB_2). Thus, the semantic of each error is univocally encapsulated in one task which is associated to only one model. Such result supports the choice made for the k value in terms of coherency and independency of the topics (Section 5.6) and the validity of the behavior models in Table 7. In the following, we discuss each error in more details. To derive the cause of an error, we compare models with error with the error-free model. To get more information on the type of error we investigate the sequences of messages characterizing a task and including the error message and analyze the classes and methods associated to the task.

E1. E1 is reported in the logs as an ERROR event. The error appears in model M_3 (Table 7) and the log message `Error reading input data (type=30001, code=8)` is the sole event message mapped into T_8. The type indicates that the error is generated within the subsystem "OFFLINE" and the code reports the generic description "Any C++ error", (ESO, 2016). The behavior model of M_3 differs from the error-free M_1 by the error tasks T_8 and the task T_6 "Set Horizon Offset". In addition, when the error occurs, the task "Pointing and Focusing" (T_1) is likelier whereas the tasks "Tuning analog signal" (T_3) and "Moving antennas" (T_7) are less likely to occur than in the error-free M_1.

Thus, it appears that when the error occurs, operations to calibrate pointing and focusing have been performed as well. The method of T_8 `reportFocusModel` of class `ArrayStateBase` (Table 6) is called to generate the data model of the focus of the antennas. To perform tasks T_1 and T_6,, two methods of the class `ArrayMountControllerImpl` are specifically called: `setSubreflectorPositionOffset` and `setOffsetPrivate`. According to the manuals, again such methods refer to the calibration of the horizon of the antennas for focusing and pointing.

Summarizing, we can say that the error in T_8 must be related to the data model used to focus the antennas and its subreflectors' offset calibration. This has been verbally confirmed by the ALMA software engineer: "the error code refers to problems in the *DataCapturer* subsystem (Fig. 2) and concerns invalid or missing data of antennas' focus."

E2. E2 is reported in the logs as an ERROR event. The error appears in group two (Table 7) and the log message `Operation Timeout (type=10000, code=14)` is the sole event message mapped into T_5 "Move ACD" Table 3. The type value refers to the subsystem "CONTROL" and the code 14 is described as "Any TICS TCorr error". As we are analyzing logs of the CONTROL subsystem the type does not add much information and we were not able to find a more verbose description in the manuals. According to the ALMA manuals, ACD is the mechanical arm that measure the ambient or universe temperature for an observation of the universe. The pattern *E2W* in T_5 (Table 3) indicates that the E2 is followed by a message of a WARNING event (annotated as W). The warning message is more informative:

Timed out ACD to get into position. Not shutdown the observation. Data will be flagged & subsequent power level optimizations may be incorrect

Thus, the warning concerns the timeout of the mechanical arm ACD to return to a certain position and the corruption of power level optimization data. As declared in the warning message the error does not stop the execution of the SBs. As in the case of E1, the behavior model M_2 differs from the free-of-errors model M_1 by tasks T_2 "Calibrate Atmosphere Data" and T_6 "Set Horizon Offset."

Thus, it appears that when the error occurs in T_5, operations to calibrate atmosphere data have been performed as well. The methods *LocalOscillatorImpl::waitForACD* and *AntennaCallback-Base::waitForEnoughResponses* are solely associated to topic T_5 (Table 6). In addition, the pattern of messages we found for T_5 *BD E2W* (Table 3) associates the timeout error to the movement of ACD from the "AMBIENT" to the "HOT" load. According to the ALMA manuals, this indicates that the timeout is related to the temperature measurement for the observation ("HOT" load) and the call back of the mechanical arm during such measurement.

Summarizing, the error is due to a timeout of the mechanical arm ACD when it moves to the position in which it measures the temperature for universe's observation.

E3. E3 in group four is the error we chose in the JIRA database. The error is about the initialization of a meta-data model that later is saved in the database. It is reported in JIRA data base and not in the logs. Our approach detects it because at the time stamp of an occurrence of such error in the selected time window, the SB immediately stops and only task T_9 is performed. Inspecting the logs, we could see that in less than 30 seconds, the operator was actually able to restart the execution with SB_{14}, on the same array of antennas, $A007$ and with no such an error. Therefore, this error seems not to be systematic and, as such, it can be classified as critical as there is no indication of when it will occur again. In the bug report in JIRA, software engineers came to the same conclusion after few days of discussions and trials:

It is clear that during January there was only one instance lasting more than 100 seconds, while in the last week there have been nine of such slow initializations.

7. Lessons learned in mining a large corpus of log events

In this section, we briefly summaries the how we overcome new and old challenges of mining log to extract system behavior.

Short texts. To overcome the lack of vocabulary of short text messages, we implemented two strategies: 1) the behavior of a system is modelled by hidden topics in sequences of event messages and not by individual message and 2) topics are labelled using patterns of such messages not bag of words.

Context investigation. A thorough context analysis and validation of the results with domain experts was crucial for the application of our approach starting from the definition of the sample that better suited our research goal, Section 4.2.

Sequences of events. This work adopts the approach in (Russo et al., 2015) that uses sequences of events as input for a machine learner. In this study, sequences of event messages are used to as input for the LDA analysis, whereas recurring sequences of messages are used to label the resulting telescope's tasks.

Observation time window. In this work, a deep context analysis has guided the choice of the samples to select a time period during which the telescope is fully operational and does not undergo any maintenance activity.

Fully-automated log mining. The approach of this work automated a large part of the manual activities of log mining performed at ALMA. Two steps of our approach are still manual: selecting the number of topics k and labelling message patterns. Any of these manual activities can be automated a bit more though losing in accuracy. For example, the choice of the number of topics k can be automated by using the perplexity function leaving only the interpretation of the resulting perplexity curve to the subjectivity of the researcher. Yet, this approach does not assure accuracy in terms of topics coherency and independency (Layman et al., 2016). In this work, we preferred to introduce a more accurate approach that requires the manual inspection of the distribution of the messages over topics to obtain topics with internal coherent semantic.

Manual labelling message patterns can be replaced by automatic extraction of topic labels from manuals or online articles like in the work of Lan *et al.* (Lau et al., 2011). The approach in (Lau et al., 2011) is not fully automated though as it requires a ground truth derived by manual inspection of candidate online articles.

Fault localization. Our approach is able to associate low-level code components to tasks and SBs can be reconstructed as sets of tasks. When one of such tasks contains an error, the code components of the SBs with the error can be retrieved. When the error can effectively be associated to a system malfunction, such information can be exploited by the software engineers as starting point for the fixing process and replace manual localization of the issue as in the current practice at ALMA. Of course, the malfunction can be originated in files and methods not directly included in the information of the log events and nested deeper in the code. The analysis of the execution traces can help to get access to such nested code (Tan et al., 2008), but they are not available in the logs mined in this work.

Incomplete information and identical timestamp. The approach in this work clusters the information carried by individual events and reconstructs the latent semantic of such clusters. In this way, incomplete information in a message of a single event is augmented by the information carried by the messages of the same cluster taming down the negative effects of incomplete information. Likewise, identical time stamps have little to no effect on our findings as the time stamp is used only to distinguish SBs by means of their starting and ending events.

8. Threats to validity

Given the explorative nature of this work, the relevant class of threats falls under *construct validity*.

Firstly, the event messages may still contain a vocabulary specific to non-ordinary activities of the system (e.g., ad hoc maintenance) although we accurately selected the logs during the Observation phase of the telescope. Thus, to understand whether this is the case, the vocabularies of the messages of the testbed and the validation sample have been compared. The testbed vocabulary contains 114 distinct terms, which are all contained in 169 terms of the validation sample vocabulary. Many of the remaining terms in the validation sample vocabulary have small differences with terms of the testbed vocabulary that text pre-processing was not able to capture (e.g., "Focus", "FocusX"). Thus, the two vocabularies are not much different.

The ALMA software engineer also confirmed that the vocabulary during the Observation phase of the ALMA telescope is in fact homogeneous as it has to follow the technical standards of the Observation cycle (Asayama et al., 2017)), although it leaves engineers enough room to include their own terms especially in case of unexpected or erroneous behavior. On the other hand, this flexibility is not considered a good practice and, in part, motivates this work.

Secondly, mechanisms to detect and explain anomalies in logs still suffer of false positives as the information included in individual log events can be missing or not completely accurate (Oliner et al., 2012; Cinque et al., 2013), ChuahEtAl2010). In our work, we tamed this issue by clustering event messages into topics so that the information in individual error events is augmented with the one in events of the same cluster. We further were able to build behavior models that do not associate an error to specific context of operation (array of antennas) or time frame (SB execution).

Another issue is that labelling topics is manual process that implies subjective bias. In our work, we propose a method that increase the expressivity of the information carried by topics and help researchers or practitioners label the topics.

Finally, the LDA machinery have its own limitations. In LDA, each document is represented as a bag of words. One problem of this representation is that the model is responsible for figuring out which dimensions in the document vectors are semantically related. Leveraging information on how words are semantically correlated to each other may improve a model's performance and this is what word embeddings do and may be matter of future work.

Generalizability is a typical class of threats for explorative studies: logs and the information they carry might be specific of the ALMA context. For example, the identification of sequences of events to be used as LDA documents might not be straightforward as events or their messages might not contain clear indication of a sequence's start or end. To define the sequences, a deeper context analysis and informative research must be performed (Russo et al., 2015)

9. Conclusion

In this work, we propose a method that exploits the information contained in log events to reconstruct system behavior as set of telescope's tasks. The work uses LDA analysis to identify the number of such tasks and a pooling schema improvement based on pattern recognition to label such topics. The application of our method illustrates how to mine about 2000 events and reconstruct the tasks of 16 sequences of events each describing an observation of the ALMA telescope. The method is also able to identify anomalies in logs, consistently over sequences and derive the semantic of such malfunctions by analyzing the tasks and the code components involved in the tasks of an SB. The overall approach aims at reducing to the minimum the manual activities for log mining that are currently adopted at ALMA. With this work, we have also been able to discuss some of the known challenges in log mining. The experience we gather has been then summarized in lessons learned.

In future work, we will explore how to increase further the automation of the proposed method. In particular, we will study the feasibility to replicate the work of Lau *et al.*(Lau et al., 2011) in the ALMA context. In such work, the authors propose an approach that parses online, textual references (e.g., wikipedia articles) to derive candidate labels, defines the ground truth with domain experts, and uses supervised vector machines to associate labels to topics (Lau et al., 2011). In principle, such approach can be applied to the ALMA logs. To understand whether such replication makes any sense, we took the bag of words for Topic 1 in Table 4 and searched with google engine the words together with the term "ESO" (European Southern Observatory) to identify the candidates' textual references as in (Lau et al., 2011). Interestingly, the first five pages we found speak about the "tuning of image signal" and the fifth page is the ALMA manual of the 2017 cycle (the telescope's cycle from which the logs of this work were collected). "Tuning image signal" is the label of topic T_1 indeed, Table 5. Such finding is promising and the technique in (Lau et al., 2011) has a good potential to be replicated with our logs by simply replacing the ground truth with the labels that we found with your approach, Table 5.

Acknowledgment

This work has been partially supported by the GAUSS Italian research project (funded by MIUR, PRIN 2015, Contract: 2015KWREMX) and DEBASS research project (funded by Free University of Bolzano).

References

Agrawal, A., Fu, W., Menzies, T., 2018. What is wrong with topic modeling? and how to fix it using search-based software engineering. Information and Software Technology 98, 74 – 88.

ALMA, 2017. Alma glossary.
URL `https://science.nrao.edu/facilities/alma/aboutALMA/Technology/ALMA_Computing_Memo_Series/draft/Glossary.pdf`

ALMA-Newsletter, 2012. Alma calibrations. ALMA Newsletter.

Andrews, J. H., Zhang, Y., July 2003. General test result checking with log file analysis. IEEE Transactions on Software Engineering 29 (7), 634–648.

Asayama, S., Biggs, A., de Gregorio, I., Dent, B., Francesco, J. D., Fomalont, E., Hales, A., Hibbard, J., Marconi, G., Kameno, S., Vilaro, B. V., Villard, E., Stoehr, F., 2017. Alma cycle 5 technical handbook. Tech. Rep. 978-3-923524-66-2, ALMA partnership.

Asayama, S., Biggs, A., de Gregorio, I., Dent, B., Francesco, J. D., Fomalont, E., Hales, A., Humphries, E., Kameno, S., Mueller, E., Vilaro, B. V., Villard, E., Stoehr, F., 2016. Alma cycle 4 technical handbook. Tech. rep., ALMA partnership.

Benoit, K., Muhr, D., Watanabe, K., Muhr, D., 2017. Smart.
URL `https://cran.r-project.org/web/packages/stopwords/stopwords.pdf`

Bertero, C., Roy, M., Sauvanaud, C., Tredan, G., Oct 2017. Experience report: Log mining using natural language processing and application to anomaly detection. In: 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE). pp. 351–360.

Blei, D. M., Ng, A. Y., Jordan, M. I., 2003. Latent dirichlet allocation. Journal of Machine Learning Research.

Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C., Blei, D. M., 2009. Reading tea leaves: How humans interpret topic models. In: Proceedings of the 22nd International Conference on Neural Information Processing Systems. NIPS'09. Curran Associates Inc., USA, pp. 288–296.

Chuah, E., Kuo, S., Hiew, P., Tjhi, W., Lee, G., Hammond, J., Michalewicz, M. T., Hung, T., Browne, J. C., Dec 2010. Diagnosing the root-causes of failures from cluster log files. In: 2010 International Conference on High Performance Computing. pp. 1–10.

Cinque, M., Cotroneo, D., Pecchia, A., June 2013. Event logs for the analysis of software failures: A rule-based approach. IEEE Transactions on Software Engineering 39 (6), 806–821.

Damevski, K., Chen, H., Shepherd, D., Pollock, L., 2016. Interactive exploration of developer interaction traces using a hidden markov model. In: Proceedings of the 13th International Conference on Mining Software Repositories. MSR '16. ACM, New York, NY, USA, pp. 126–136.

Damevski, K., Chen, H., Shepherd, D. C., Kraft, N. A., Pollock, L., Nov 2018. Predicting future developer behavior in the ide using topic models. IEEE Transactions on Software Engineering 44 (11), 1100–1111.

ESO, 2016. Error definition.
URL http://www.eso.org/projects/alma/develop/acs/Releases/2016-06-ACS-B/APIs/errors/index.html

Featherstun, R. W., Fulp, E. W., 2010. Using syslog message sequences for predicting disk failures. In: Proceedings of the 24th International Conference on Large Installation System Administration. LISA'10. USENIX Association, Berkeley, CA, USA, pp. 1–10.

Fronza, I., Sillitti, A., Succi, G., Terho, M., Vlasenko, J., 2013. Failure prediction based on log files using random indexing and support vector machines. Journal of Systems and Software 86 (1), 2–11.
URL https://doi.org/10.1016/j.jss.2012.06.025

Fu, S., Xu, C.-Z., Nov. 2010. Quantifying event correlations for proactive failure management in networked computing systems. J. Parallel Distrib. Comput. 70 (11), 1100–1109.

Gadler, D., Mairegger, M., Janes, A., Russo, B., 2017. Mining logs to model the use of a system. In: 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2017, Toronto, ON, Canada, November 9-10, 2017. pp. 334–343.

Gazzola, L., Micucci, D., Mariani, L., 2019. Automatic software repair: A survey. IEEE Transactions on Software Engineering 45 (1), 34–67.

Gil, J. P., Reveco, J., Shen, T. C., 2016. Operational logs analysis at alma observatory based on elk stack. In: Guzman, G. C. J. C. (Ed.), Proc. SPIE 9913, Software and Cyberinfrastructure for Astronomy IV. Vol. 9913.

Goldstein, M., Raz, D., Segall, I., Oct 2017. Experience report: Log-based behavioral differencing. In: 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE). pp. 282–293.

Gordon, A. D., 1999. Classification (2nd ed.). Vol. 82 of Monographs on statistics and applied probability. Boca Raton, FL, : Chapman and Hall/CRC.

Griffiths, T. L., Steyvers, M., 2004a. Finding scientific topics. Proceedings of the National Academy of Sciences 101 (suppl 1), 5228–5235.

Griffiths, T. L., Steyvers, M., 2004b. Finding scientific topics ;. PNAS 2004 101 (suppl 1), 5228–5235.

Gruen, B., Hornik, K., 2011. topicmodels: An r package for fitting topic models. Journal of Statistical Software 40, 1–30.

Guille, A., Favre, C., Aug 2014. Mention-anomaly-based event detection and tracking in twitter. In: 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014). pp. 375–382.

Hindle, A., Barr, E. T., Su, Z., Gabel, M., Devanbu, P., 2012. On the naturalness of software. In: Proceedings of the 34th International Conference on Software Engineering. ICSE '12. IEEE Press, Piscataway, NJ, USA, pp. 837–847.
URL http://dl.acm.org/citation.cfm?id=2337223.2337322

Hindle, A., Bird, C., Zimmermann, T., Nagappan, N., 20. Do topics make sense to managers and developers? Empirical Software Engineering 2, 479–515.

Jiang, B., K. Yoshigoe, A. H., Yuan, J., He, Z., Xie, M., Guo, Y., Prosperi, M., Salloum, R., Modave, F., 2016. Mining twitter to assess the public perception of the "internet of things". PLoS one 11 (7).

Jiang, Z. M., Avritzer, A., Shihab, E., Hassan, A. E., Flora, P., June 2010. An industrial case study on speeding up user acceptance testing by mining execution logs. In: 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement. pp. 131–140.

Jiang, Z. M., Hassan, A. E., Hamann, G., Flora, P., 2008. An automated approach for abstracting execution logs to execution events. Journal of Software Maintenance and Evolution: Research and Practice 20 (4), 249–267.
URL https://doi.org/10.1002/smr.374

Khodabandelou, G., Hug, C., Deneckère, R., Salinesi, C., 2014. Unsupervised discovery of intentional process models from event logs. In: Proceedings of the 11th Working Conference on Mining Software Repositories. MSR 2014. ACM, New York, NY, USA, pp. 282–291.

Koltcov, S., Koltsova, O., Nikolenko, S., 2014. Latent dirichlet allocation: Stability and applications to studies of user-generated content. In: Proceedings of the 2014 ACM Conference on Web Science. WebSci '14. pp. 161–165.

Lau, J. H., Grieser, K., Newman, D., Baldwin, T., 2011. Automatic labelling of topic models. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (HLT '11), Vol. 1. Association for Computational Linguistics, Stroudsburg, PA, USA. pp. 1536–1545.

Lau, J. H., Newman, D., Baldwin, T., 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In: In Proc. of the European Chapter of the Association for Computational Linguistic (EACL).

Layman, L., Nikora, A. P., Meek, J., Menzies, T., 2016. Topic modeling of NASA space system problem reports: research in practice. In: Proceedings of the 13th International Conference on Mining Software Repositories, MSR 2016, Austin, TX, USA, May 14-22, 2016. pp. 303–314.

Mariani, L., Pastore, F., Nov 2008. Automated identification of failure causes in system logs. In: 2008 19th International Symposium on Software Reliability Engineering (ISSRE). pp. 117–126.

Mehrotra, R., Sanner, S., Buntine, W., Xie, L., 2013. Improving lda topic models for microblogs via tweet pooling and automatic labeling. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '13. ACM, New York, NY, USA, pp. 889–892.

Menzies, T., Marcus, A., Sept 2008. Automated severity assessment of software defect reports. In: 2008 IEEE International Conference on Software Maintenance. pp. 346–355.

Nguyen, A. T., Nguyen, T. T., Nguyen, T. N., Lo, D., Sun, C., 2012. Duplicate bug report detection with a combination of information retrieval and topic modeling. In: Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. ASE 2012. ACM, New York, NY, USA, pp. 70–79.

O'Callaghan, D., Greene, D., Carthy, J., Cunningham, P., 2015. An analysis of the coherence of descriptors in topic modeling. Expert Systems with Applications 42 (13), 5645 – 5657.

Oliner, A., Ganapathi, A., Xu, W., Feb. 2012. Advances and challenges in log analysis. Commun. ACM 55 (2), 55–61.

Oliner, A. J., Stearley, J., 2007. What supercomputers say: A study of five system logs. In: The 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2007, 25-28 June 2007, Edinburgh, UK, Proceedings. pp. 575–584.

Porter, M., Boulton, R., Bouchet-Valat, M., 2019. Snowballc.
URL https://CRAN.R-project.org/package=SnowballC

Rouillard, J. P., 2004. Real-time log file analysis using the simple event correlator (sec). In: LISA '04 Proceedings of the 18th USENIX conference on System administration, Atlanta, GA, November 14 - 19, 2004. pp. 133–150.

Russo, B., Succi, G., Pedrycz, W., 2015. Mining system logs to learn error predictors: a case study of a telemetry system. Empirical Software Engineering 20 (4), 879–927.

Salfner, F., Schieschke, M., Malek, M., April 2006. Predicting failures of computer systems: a case study for a telecommunication system. In: Proceedings 20th IEEE International Parallel Distributed Processing Symposium. pp. 8 pp.–.

Simon, K., Malcolm, C., 2006. An Introduction to Knowledge Engineering. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Sommer, H., Chiozzi, G., Zagar, K., Voelter, M., 2004. Container-component model and xml in alma acs. Vol. 5496. pp. 219–229.

Tan, J., Pan, X., Kavulya, S., Gandhi, R., Narasimhan, P., 2008. Salsa: Analyzing logs as state machines. In: Proceedings of the First USENIX Conference on Analysis of System Logs. WASL'08. USENIX Association, Berkeley, CA, USA, pp. 6–6.

van der Aalst, W. M. P., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G., Weijters, A. J. M. M., Nov. 2003. Workflow mining: A survey of issues and approaches. Data Knowl. Eng. 47 (2), 237–267.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A., 2012. Experimentation in Software Engineering. Computer Science. Springer Berlin Heidelberg.

Xia, X., Lo, D., Ding, Y., Al-Kofahi, J. M., Nguyen, T. N., Wang, X., March 2017. Improving automated bug triaging with specialized topic model. IEEE Transactions on Software Engineering 43 (3), 272–297.

Zhao, W., J Chen, J., Perkins, R., Liu, Z., Ge, W., Ding, Y., Zou, W., 09 2015. A heuristic approach to determine an appropriate number of topics in topic modeling 16, S8.

Zou, L., Song, W. W., 2016. Lda-tm: A two-step approach to twitter topic data clustering. In: 2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Chengdu, 2016. pp. 342–347.