The European Southern Observatory

Log Analysis as an Operational Tool at Paranal Observatory

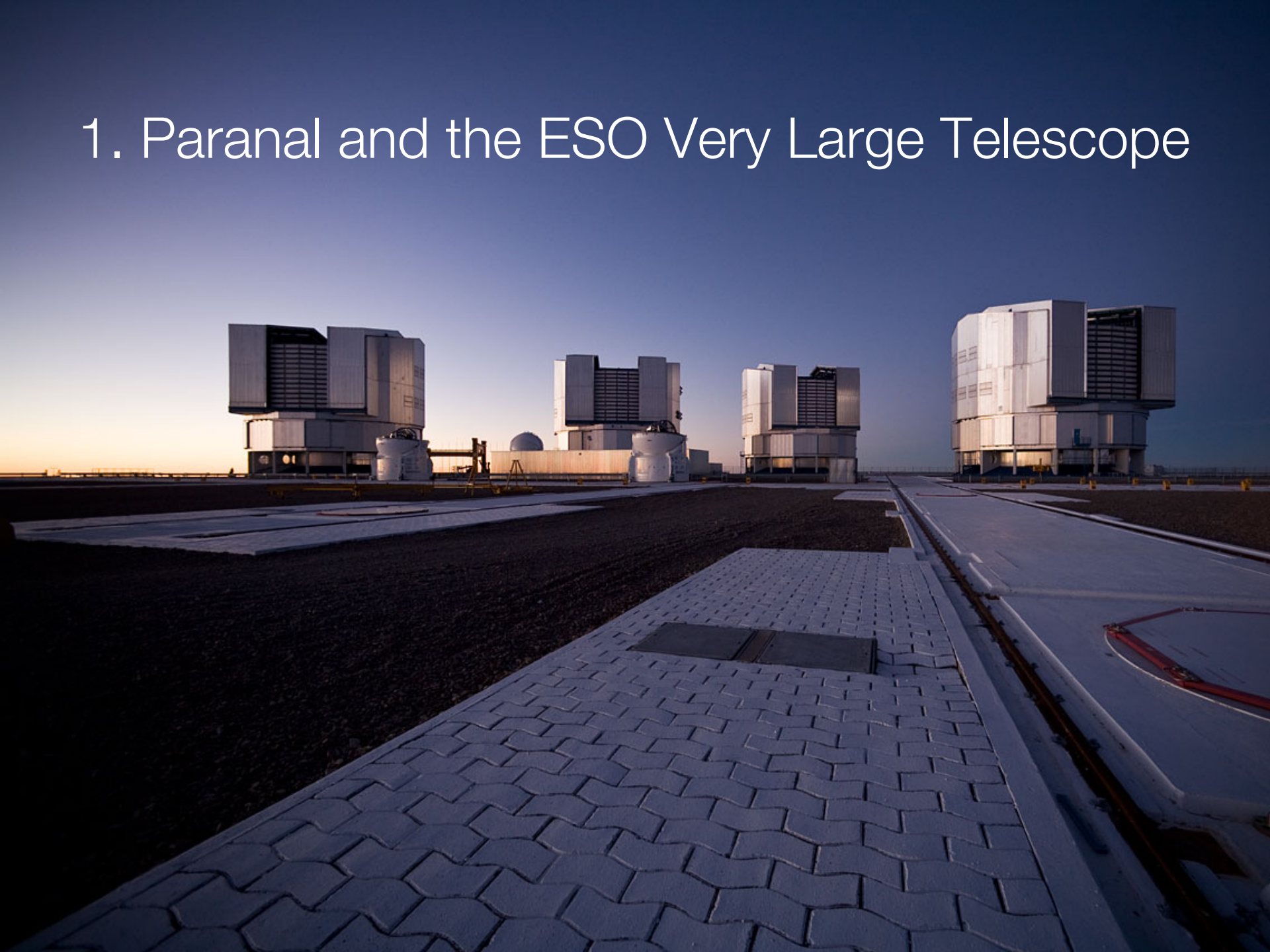
SciOps 2022: Artificial Intelligence for Science and Operations in Astronomy

Juan Pablo Gil – jgil@eso.org
Paranal SW Group / Paranal Datascience Group

Overview

1. Paranal and VLT
2. Log Infrastructure
3. Log Analysis Basics
4. Examples
5. Conclusions

1. Paranal and the ESO Very Large Telescope



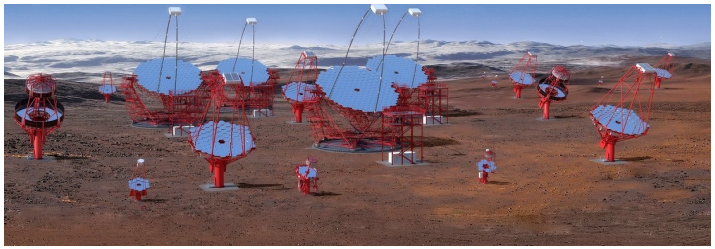
Current Paranal Configuration

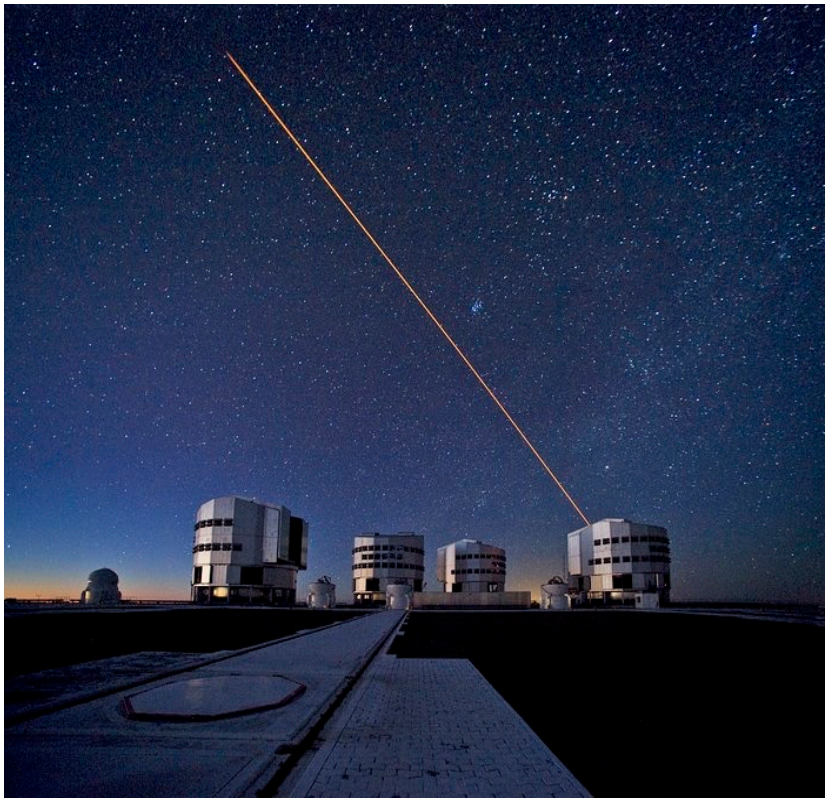
VLT (UT & VISTA):
- 5 telescopes
- 14 instruments
- AO and AOF

VLT (VLTI):
- 4 telescopes
- 8 AO Systems (UT & AT)
- 3 instruments

Supporting Systems:
- Power Supply
- Cooling Supply
- Coating Units

Evolution of Paranal Configuration





Credit: Gabriel Pérez Diaz, IAC / Marc-André Besel, CTAO

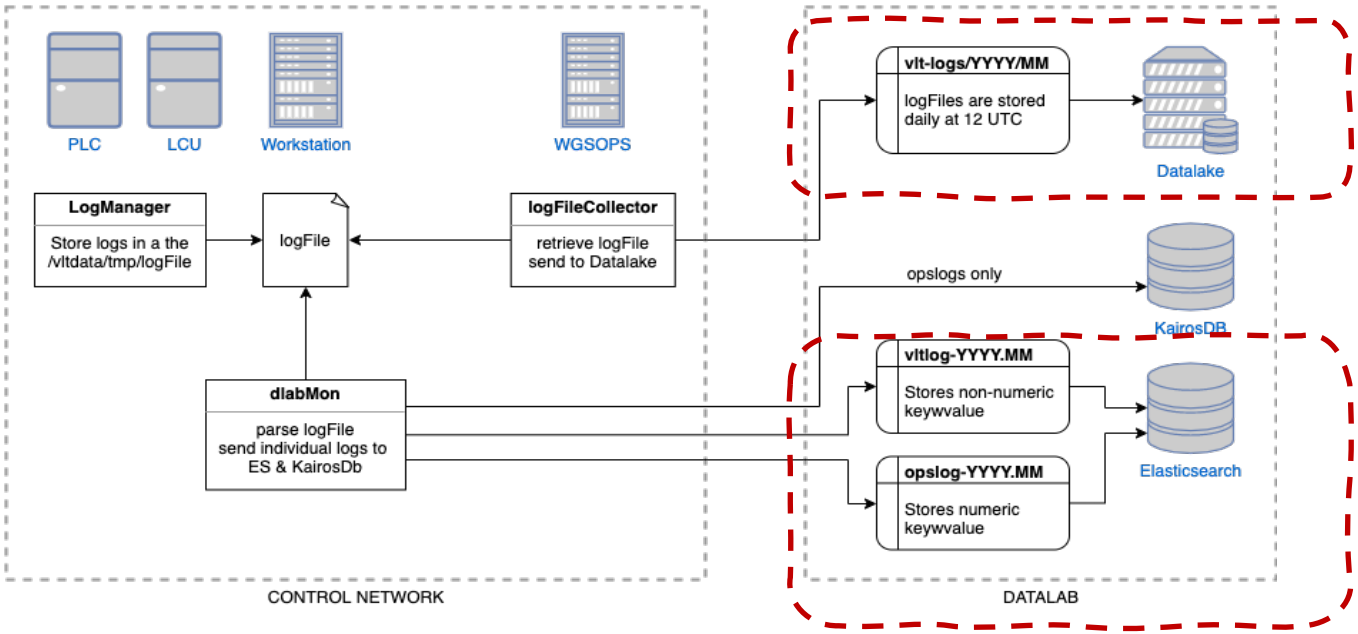
Paranal Software Group

- Deploy & Maintain Core Ops Software
  - VLTSW as common middleware.
  - Telescope SW
  - Instrument SW
- Support new developments
  - New Instruments
  - Ancillary systems
- Troubleshooting
  - In a daily basis
  - Done mainly analyzing software logs

2. Log Infrastucture at Paranal

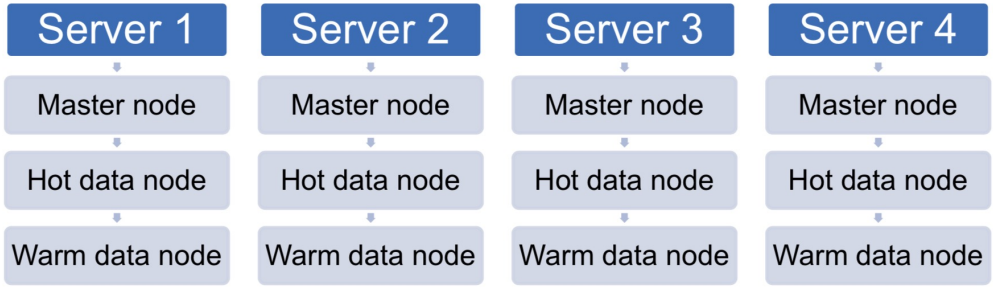
Log Collection

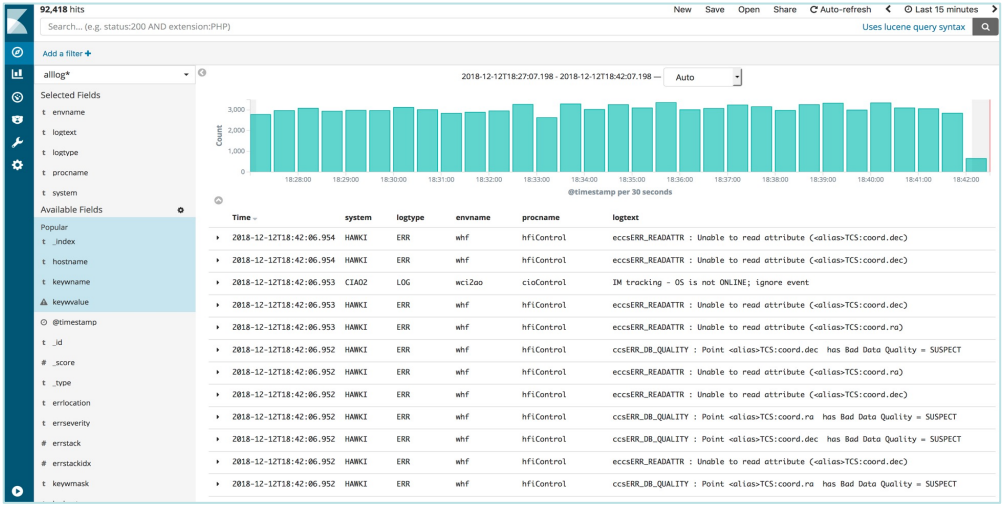
- VLTSW provides a common log functionality
  - But in text files local to each system (+40)
- Logs from all systems are sent to Paranal Datalab
  - Raw files inside the Datalake
  - Elasticsearch



Log Storage and Query

- Backend in Elasticsearch
- Kibana for visualization
- Industry de facto standard
- Off the shelf
- Distributed
- High speed queries





Logs in Numbers

- +40 sources generating VLT Software logs
  - Sent from VLT Software to Elasticsearch in real time
- Insertion rate
  - Average single log line size: ~200 bytes
  - Average at operation: 150 logs/second (0.03 MB/s)
  - Observed Peak: 70,000 logs/second (13.4 MB/s)
- Indices
  - One index per month per type:
    - vltlog-YYYY.MM
    - opslog-YYYY.MM
  - ~30 GB/month per type
- Historical data
  - 15 years in raw files
  - 5 years in Elasticsearch, ~10 TB of logs

3. Log Analysis Basics

Types of Analysis

- Log Based Tools
- Find nearest executions
- Probability of execution error
- Event Labelling:
  - Error Markers
  - Important Event (explain clustering)
  - Suspicious Events
  - Strange Events
- Execution Simulation
- Incomplete Execution detection

Steps for Log Analysis

Dataset Generation

- High Level Task
  - Observation
  - Template Exec.
  - PRESET
  - SELINS
- Split in Traces
- Local CSV Files

Pre-processing

- Cleansing: remove bad traces
- Tokenization (equivalent to stemming, etc, in NLP)
- Vocabulary generation

Event Representation

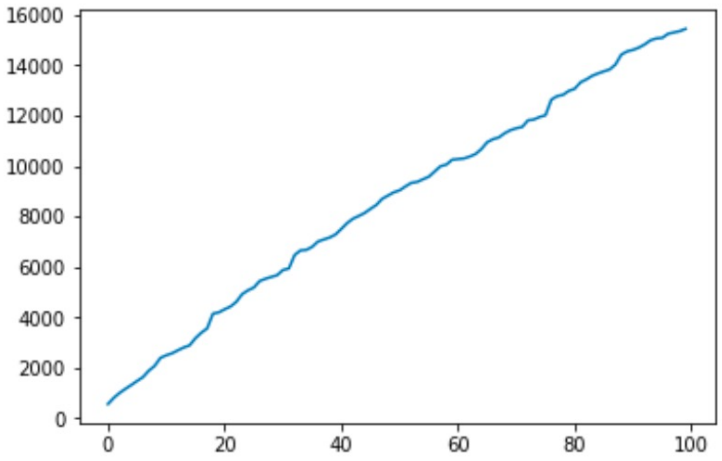
- Time series
- Tokens
- Bag of Words (BoW)
- One Hot BoW
- Successor Graphs

Technique

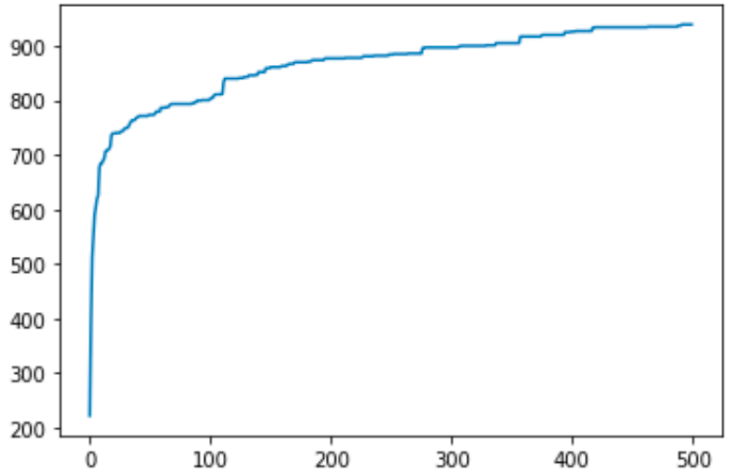
- Time series analysis
- Naive Bayes
- Cluster detection:
  - K-Means
  - SVM
  - T-SNE
  - UMAP
- Topic Modelling
  - LDA
- Deep Learning
  - Transformers
- Sequence Detection

Pre-processing

- Tokens are unique entities by removing variable parts in logs
- The set of unique tokens is called the **vocabulary**
  - Reasonable Size
  - Same action, same token
  - Should converge
- RegExp as Basic approach
- An Optimised method was needed to reach convergence



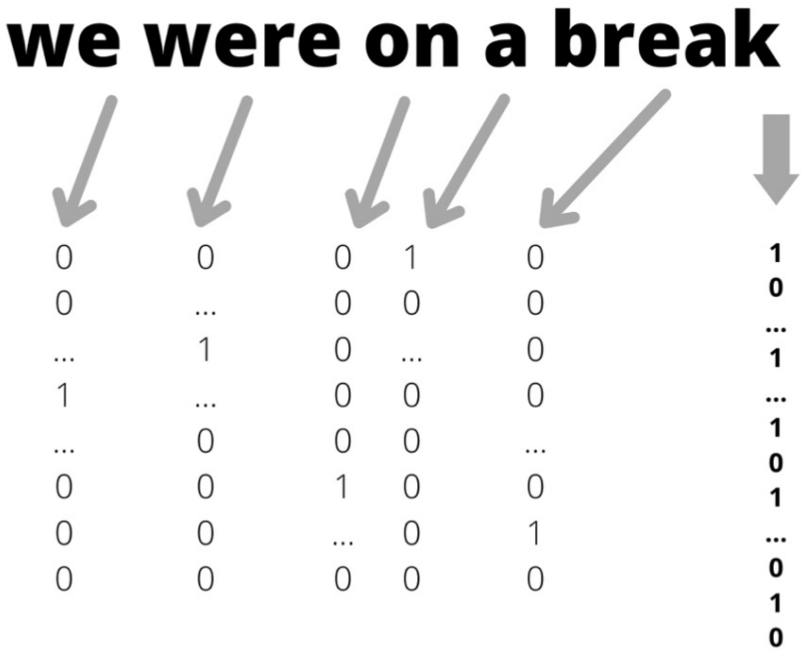
(a) BasicColor



(b) Optimised color

Bag of Words Methodology

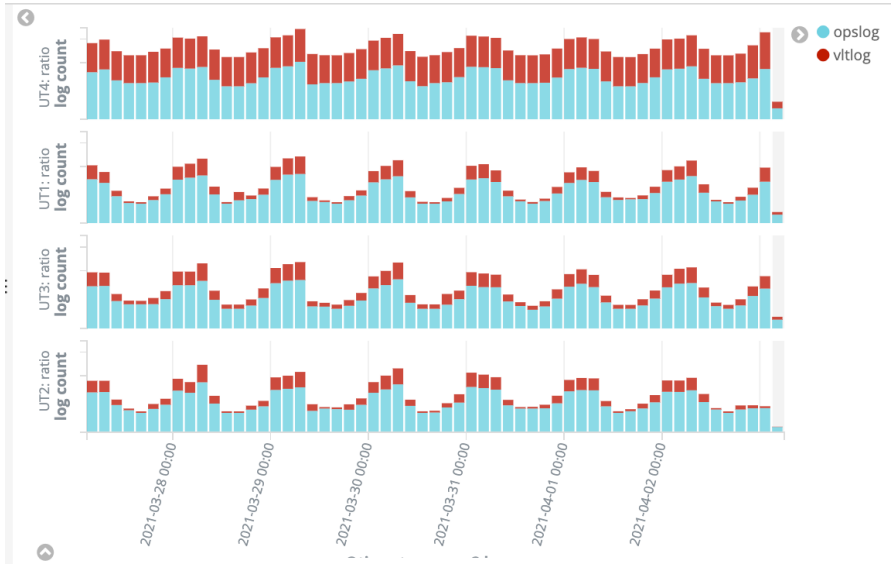
- Colorise the sequence, i.e., map the execution to a vocabulary of unique tokens
- Transform the executions to a bag-of-words vector
- Define a distance. Notice it can be a binary one.
- Usage:
  - Nearest executions
  - Important log events
  - Classification (after dimensionality reduction)



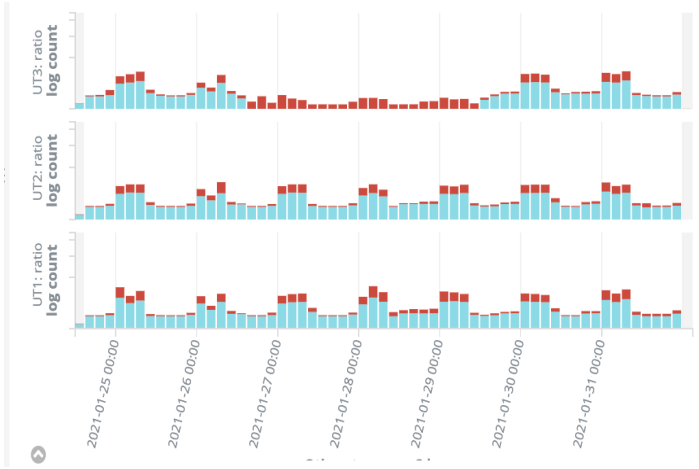
4. Examples

Timeseries: anomaly detection

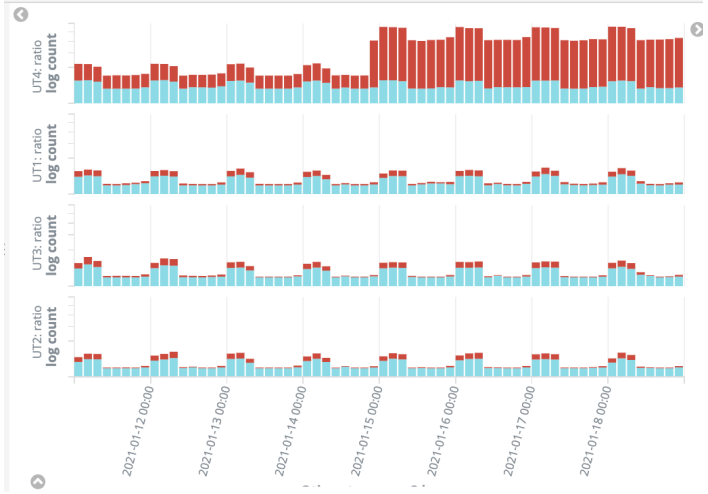
A) Normal Week



B) Fault State: UT3 not generating ops logs (blue)



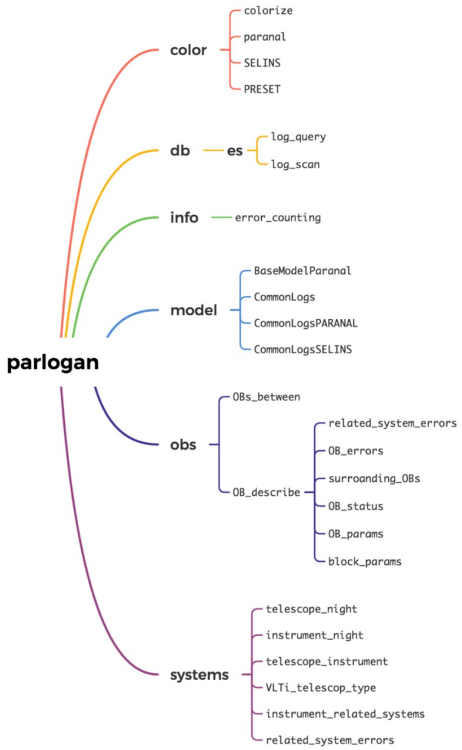
C) Anomaly: Abnormal UT4 increase in vlt logs (red)



PARLOGAN Python library

Goal: Provide a common library for Paranal Log Analysis (ParLogAn)

Implemented in Python, relies on PANDAS



Example: disconnections during OBs

```
# Search all environment disconnection in GRAVITY
dataset = parlogan.db.es.log_scan(
    'GRAVITY AND qsemuLocal.c AND logtext: "recently disconnected"',
    "2019-05-26T03:00:00", "2019-05-29T10:00:00").sort_values(by = ['@timestamp'])
```

123 rows × 17 columns

Now restrict event disconnections just to events happens WHILE an obs was running

```
obs_start = 'GRAVITY AND logtext: "OB started at"'
obs_end = 'GRAVITY AND logtext: ("OB finished" "OB execution aborted")'

dataset_filtered = parlogan.in_between(dataset, obs_start, obs_end)
```

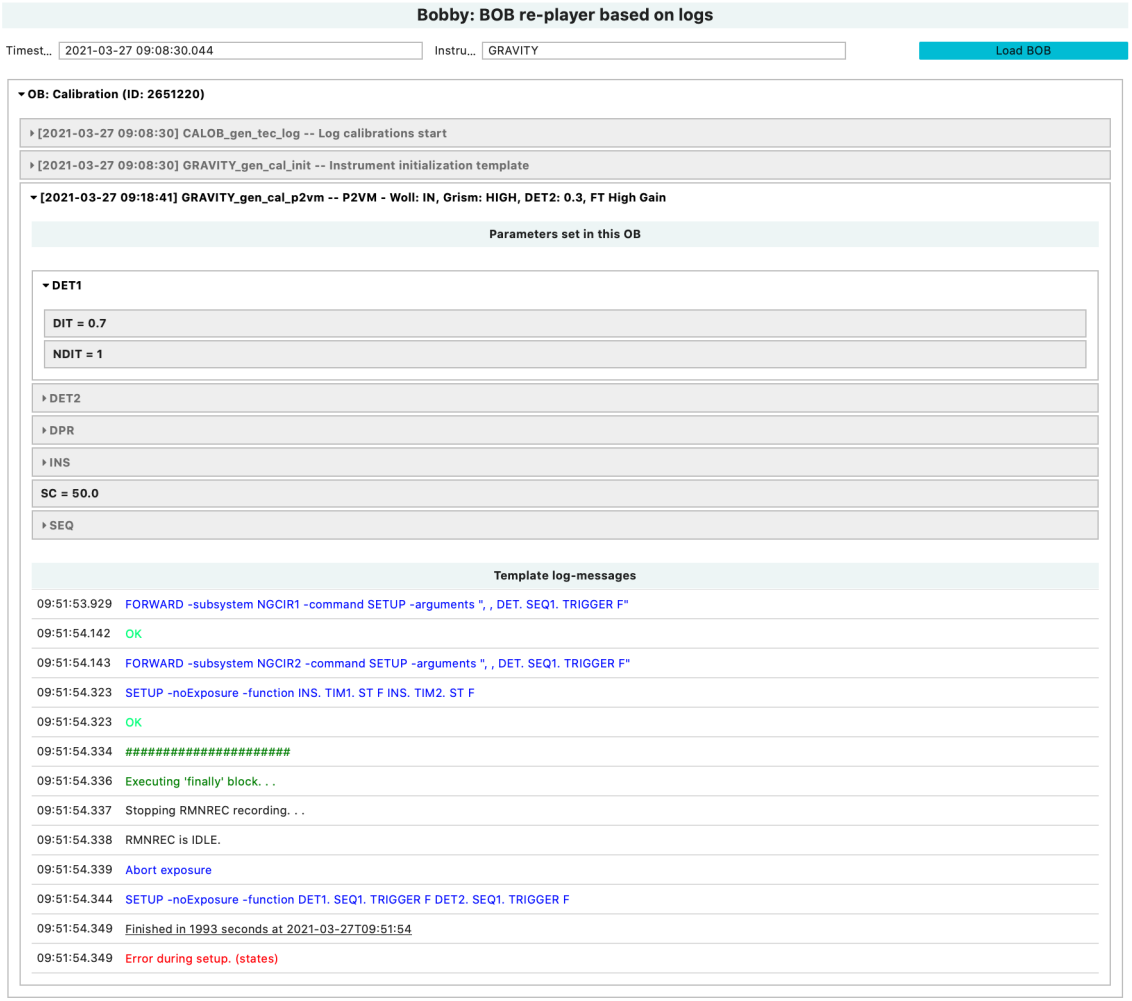
65 rows × 17 columns

	@timestamp	logtype	system	loghost	hostname	envname	logtext
121	2019-05-29T01:55:35.650	LOG	GRAVITY	wgv	wgv	wgv	qsemuLocal.c(318): environment lgvmet recently...
122	2019-05-29T01:55:35.650	LOG	GRAVITY	wgv	wgv	wgv	qsemuLocal.c 8 8166 1 W remote environment rec...
67	2019-05-29T01:55:35.652	LOG	GRAVITY	wgv	wgv	wgv	qsemuLocal.c 8 8169 1 W remote environment rec...
68	2019-05-29T01:55:35.655	LOG	GRAVITY	wgv	wgv	wgv	qsemuLocal.c 8 8190 1 W remote environment rec...
89	2019-05-29T01:55:35.656	LOG	GRAVITY	wgv	wgv	wgv	qsemuLocal.c 8 8195 1 W remote environment rec...
...	...	...	...	...	...	...	...

Bobby! (BOB replayer)

Goal: re-create BOB execution and parameters used in a given observation based on logs

Written in Jupyter Notebooks
Executed in Voila

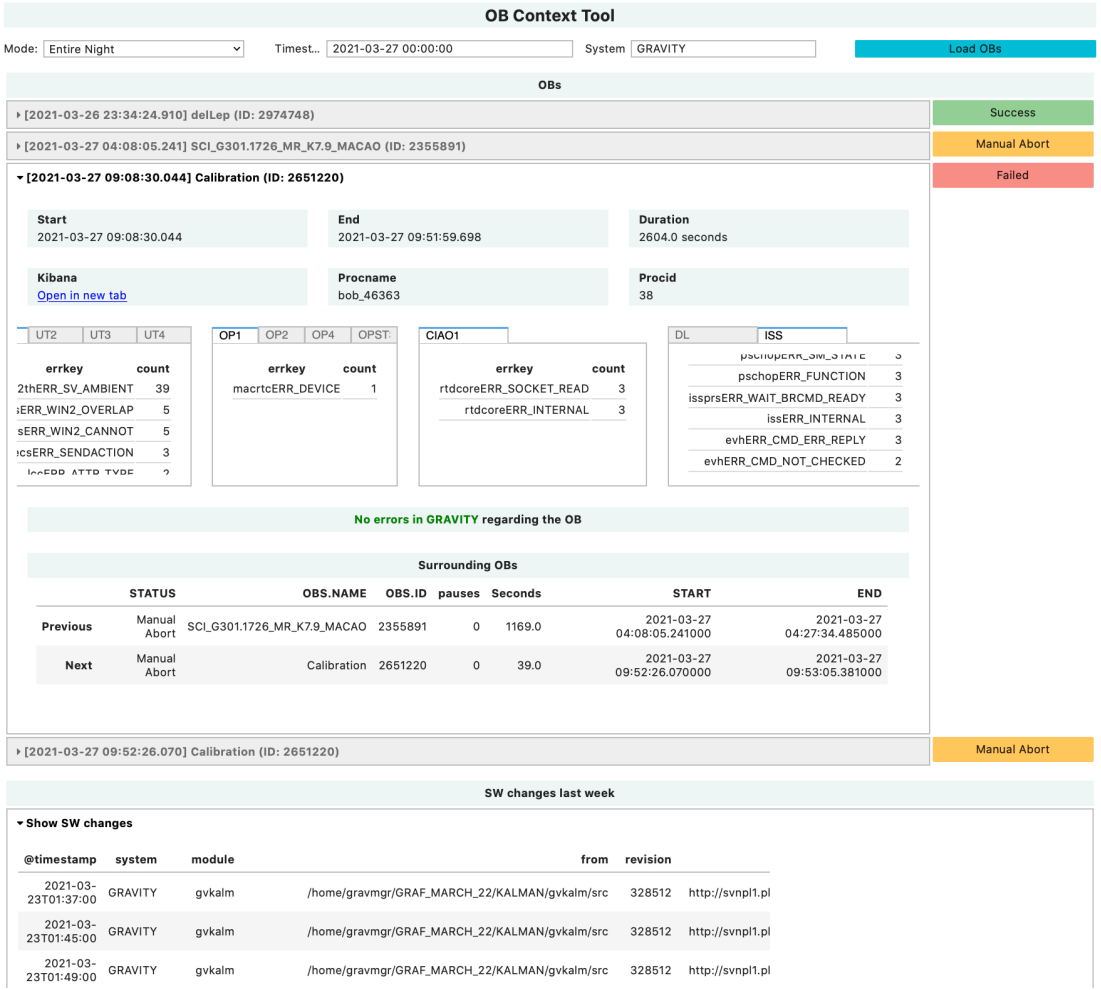


Observation Context

Goal: Provide context about

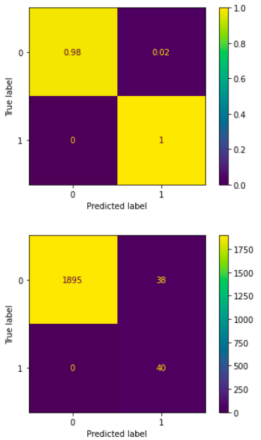
- Times
- errors in related systems
- surrounding OBs
- SW changes last week

Written in Jupyter Notebooks
Executed in Voila



Naive Bayes for Error Prediction

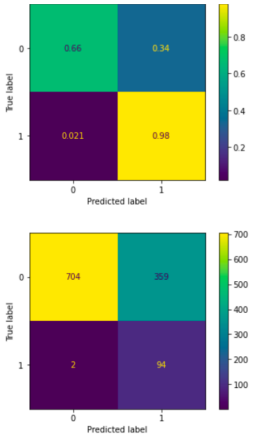
- Use Case: Template Execution
- Tokenized Events
- Small Dataset:
  - 1 day of data
  - local laptop
- Big Dataset
  - Six months of data
  - AZURE Cloud



(m) Pionier improved

Recall	Specificity	Precision	NPV	Accuracy
1.000000	0.980341	0.980720	1.000000	0.990171

(n) Stats Pionier improved



(q) Gravity improved

Recall	Specificity	Precision	NPV	Accuracy
0.979167	0.662277	0.743545	0.969502	0.820722

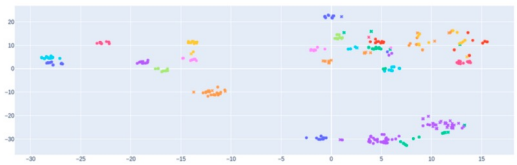
(r) Stats Gravity improved

Suspicious Events with Naive Bayes

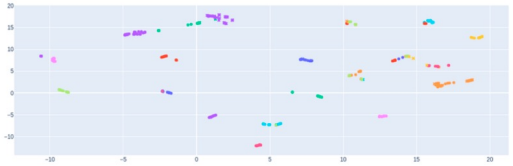
- As a sub product, the method produced a table to have a hint of events that could lead to error
- But the table must be cleaned to remove the innocent events

Token	weight
seqERR_BAD_QUALITY : "<alias>SENSOR20.last({})...	-11.400035
ccsERR_DB_QUALITY : Point <alias>SENSOR20.last...	-11.383230
... "Maintenance" OBS.START "{}" OBS.NTPL "{}"...	-8.649286
... Analysis" TPL.PRESEQ "GRAVITY_gen_tec_Pola...	-8.649286
WAIT -all (blue)	-8.235087
Setting HWP to: {} encs	-7.677115
Forward(b) SETUP to NGCIR3	-7.555012
Last Reply to 'START' from 'NGCIR3' received: ...	-7.555012
Send command 'START' '{},now' to sub-system 'N...	-7.555012
Cropping NGCIR1 failed!	-7.555012
Enable newdata newdataNGCIR3	-7.555012

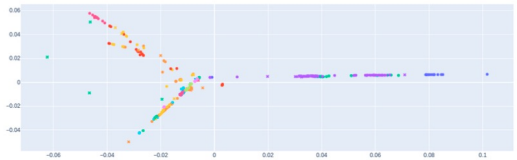
Bag of Words Results



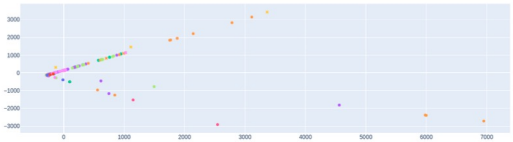
(a) t-SNE



(b) UMAP



(c) Isomap



(d) PCA

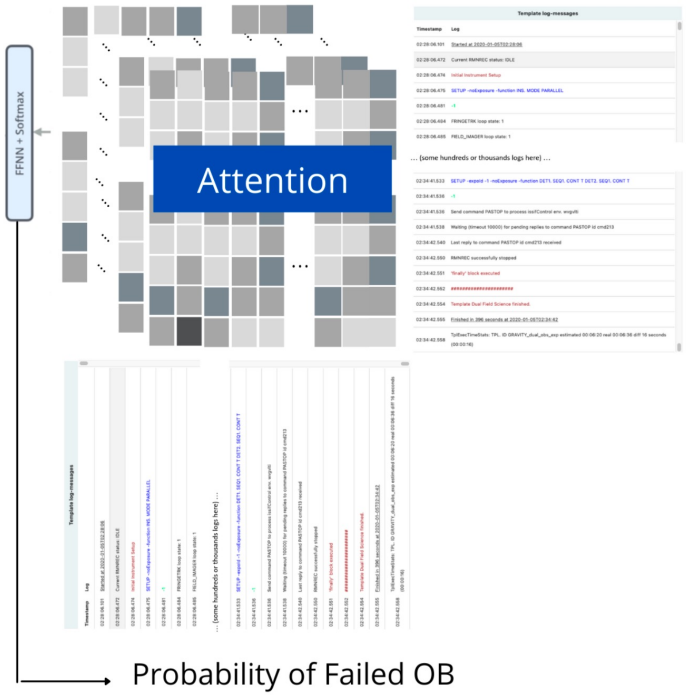
Visualisation of Gravity executions, using bag-of-words with different metrics. Colours indicate the underlying Template ID.

		precision	recall	f1-score	support
Gravity	Unsuccessful execution	0.88	0.96	0.92	370
	Correct execution	0.81	0.57	0.67	110
	accuracy			0.87	480
	macro avg	0.85	0.77	0.79	480
	weighted avg	0.87	0.87	0.86	480
Matisse	Unsuccessful execution	0.91	1.00	0.95	215
	Correct execution	0.67	0.08	0.15	24
	accuracy			0.90	239
	macro avg	0.79	0.54	0.55	239
	weighted avg	0.88	0.90	0.87	239
Pionier	Unsuccessful execution	0.97	0.99	0.98	237
	Correct execution	0.93	0.88	0.90	48
	accuracy			0.97	285
	macro avg	0.95	0.93	0.94	285
	weighted avg	0.97	0.97	0.97	285

Binary classification results (error executions) of applying SVM to a space learnt with UMAP

Transformer-based methods

- Two transformer-encoder layers
- Input and hidden dimensionality: 200
- Deep-learning framework: PyTorch
- Azure - Machine learning studio
- Masking tokens learning objective with cross-entropy loss



instrument	batch size	vocab. length	max. length	training time	cost
Gravity	8	2726	4000	35 hours 33 minutes	150 €
Matisse	16	2790	1000	3 hours 36 minutes	15 €
Pionier	16	901	500	2 hours 10 minutes	9 €

Specifications, times and estimated cost for training 200 epochs

Results with Transformers

INPUT

[' Exposure status: FINISHED (SpringGreen4)',
'ended exposure 5 of 5 (2021-01-01T05:51:33) (underlined)',
'New image: PIONIER$_O$*BS*$_F$*RINGE*001$_0$015.*fits* — — *from.Exposeof* :: *pnoseqOBS* :: *BobTPL* :: *obs*(*Blue*)′,
'WAIT -expoId -all -cond ObsEnd (blue)',
'INACTIVE (SpringGreen4)']

PREDICTION

'Template PIONIER$_o$*bs*$_c$*alibratorfinished.*′,
'Finished in seconds at (underlined)',
'TplExecTimeStats: TPL.ID PIONIER$_o$*bs*$_c$*alibrator* estimated :: real :: diff seconds (::)',

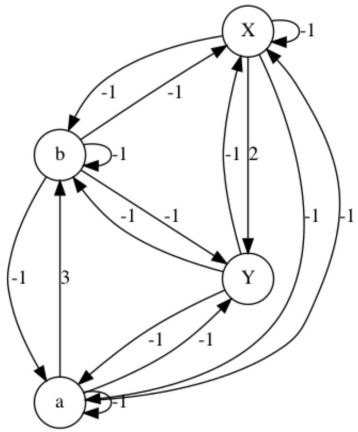
	TPL_ID	ERROR	...	seconds	length	ahead steps	ahead time
5	GRAVITY_gen_cal_init	True	...	402.0	2546	**0**	0.000
41	GRAVITY_gen_cal_dark	True	...	969.0	1419	**224**	-16.750
59	GRAVITY_gen_tec_checkMetZero	True	...	1443.0	2779	**182**	-6.103
84	CALOB_gen_tec_log	True	...	323.0	455	**36**	-0.512
86	GRAVITY_gen_cal_init	True	...	484.0	1712	**4**	-0.003

Sequence Detection

- A Sequence (*abc...yz)* is invariant across traces (log)
- Each trace is representated as a graph of successors
  - For each trace T, (a,b) is in G^T iff $(ab)^n$ is a subtrace of T with $n \geq 0$
- Graphs are combined
- Clique properties are exploited to infer the invariant sequences.
- Analytic method.

1) Log to Graph representation

```
G_T = pair_graph("abXYabXaYb")
graph(G_T)
```

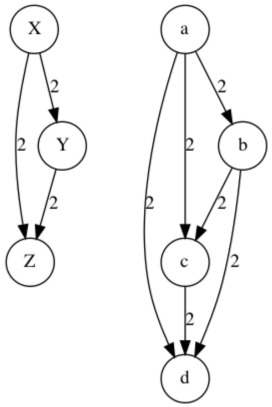


2) Properties of Graph

```
T="abcdabcdXYZXYZ"

G_T = pair_graph(T)
graph( non_negative_graph(G_T) )

print('cardinality of abcd in T = {}'.format( seque
print('cardinality of XYZ in T = {}'.format( sequen
```



3) Find Invariant Sequences

```
# Single path ABC 1234 qwxyz noise(sfd)
L = [
    list("AsBC123q41234AwBdCABCxy1234f1234AsBz
CABdC123412s34ABCAfBC123d41234ABC"),
    list(""),
    list("qw1xABC23yz41qwxyz2A3B4CAq1B23w4Cx12
Ay3B4CzAsBfC1q2f34123d4AwBdCABC1234f1234ABCABx
yzdC123412s34ABCABC123d412f34ABfC"),
    list(""),
    list("")
]
sequences_from_log(L)
```

```
{20: [['A', 'B', 'C'], ['1', '2', '3', '4']],
 5: [['q', 'w', 'x', 'y', 'z']]}
```

Found Sequences:

- ABC
- 1234
- qwxyz

Sequence Detection Results

- Transform logs to graph
- Extract SEQUENCES
- Tested with
  - 500 GRAVITY OBS
  - 3000 UT PRESETS
- Very promising results
- Requires both storage and compute power

```
================================================================================
 #0 || -------------------- in any order --------------------
 #0 || wvgrec rmiControl File transfer requested to ws:'wgv'
 #0 || wvgrec rmiControl RMNREC controlled by higher level SW via ISS (@wvgvlti:i
 #0 || wvgvlti issifControl CDP : Received command: PASTART, Buffer: wgv
 #0 || ----------------------------------------------------------
 #1 wvgrec rmiRecord Not Standalone
 #2 || -------------------- in any order --------------------
 #2 || wvgvlti issifControl CDP : Received command: STARTEX, Buffer:
 #2 || wvgvlti issifControl CDP : Succesfully completed command: PASTART, Buffer:
 #2 || wvgvlti issifControl CDP : Succesfully completed command: STARTEX, Buffer:
 #2 || wvgvlti issprsCompAct COMP_ACT: Received command: STARTEX, Buffer:
 #2 || ----------------------------------------------------------
================================================================================

================================================================================
 #0 wgv bob_ins Parallel: Start RMNREC
 #1 wgv bob_ins START -expoId {} -detId RMNREC (blue)
 #2 wgv bob_ins SETUP -function INS.TIM2.ST F INS.TIM1.ST F (blue)
 #3 wgv bob_ins SETUP -function DET1.FRAM.FORMAT cube-ext (blue)
 #4 wgv bob_ins SETUP -function DET2.FRAM.FORMAT cube-ext (blue)
 #5 || -------------------- in any order --------------------
 #5 || wgv bob_ins SETUP -expoId {} -noExposure -function INS.TIM2.ST T INS.TIM2.
 #5 || wgv bob_ins TIM Trigger start: {}
 #5 || ----------------------------------------------------------
 #6 wgv bob_ins SETUP -expoId {} -noExposure -function INS.TIM1.ST T INS.TIM1.PER
 #7 wgv bob_ins Trigger time is {}
 #8 wgv bob_ins set TPL.START to {}
 #9 || -------------------- in any order --------------------
 #9 || wgv bob_ins set INS.ROOF.POS OFFAXIS
 #9 || wgv bob_ins {}
 #9 || ----------------------------------------------------------
 #10 || -------------------- in any order --------------------
 #10 || wgv bob_ins ADDFITS done ! (SpringGreen4)
 #10 || wgv bob_ins Parallel: Start NGCIR1 at {}
 #10 || ----------------------------------------------------------
 #11 wgv bob_ins START -expoId {} -detId NGCIR1 (blue)
 #12 wgv bob_ins Parallel: Start NGCIR2 at {}
 #13 wgv bob_ins START -expoId {} -detId NGCIR2 (blue)
 #14 wgv bob_ins Send command SETST to process gvimetServer@lgvmet
 #15 || -------------------- in any order --------------------
 #15 || wgv bob_ins Parallel : WaitForAll, timeout set to : {} (blue)
 #15 || wgv bob_ins Parallel: WAIT , {}
 #15 || wgv bob_ins WAIT -expoId -all (blue)
 #15 || wgv bob_ins WaitReplyNoBlock - Waiting (timeout {}) for pending replies t
 #15 || ----------------------------------------------------------
================================================================================
```

Example: Two sequences in GRAVITY

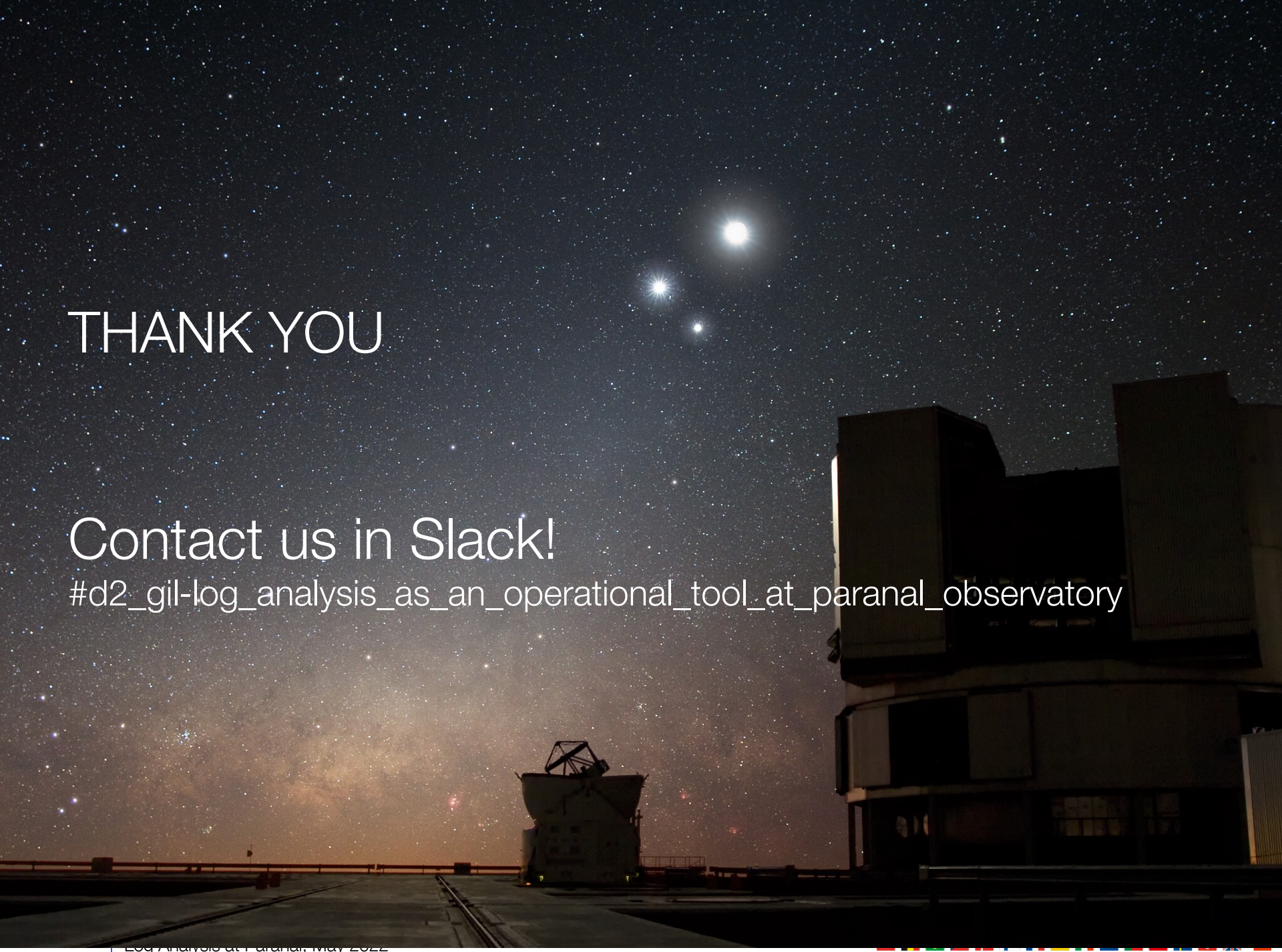
The model finds sequences even from separate subsystems: wvgrec, wvgvlti, wgv

5. Conclusions

Conclusions

- Event logs were represented as Time Series and Tokens
- Pre-preprocessing is key to have good results
- NLP techniques can be applied to log analysis
- Old school methods as Naive-Bayes and clustering works well on logs
- State of the art techniques as Transformers are very promising on software logs
- Be aware that **not every error can be explained** just from logs, example, a hardware suddenly fails then an error appears immediately.

Special thanks to internship students Camilo Carvajal, Gustavo Soto, Rubén Lagos, John Rodriguez, and Dr. Andrés Avila (UFRO).


THANK YOU
Contact us in Slack!
#d2_gil-log_analysis_as_an_operational_tool_at_paranal_observatory