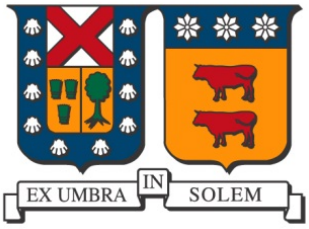
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“PROCEDIMIENTO ETL SOBRE LOS REGISTROS DE ACTIVIDAD DE LOS TELESCOPIOS UT DEL OBSERVATORIO PARANAL PARA LA GENERACIÓN DE DATASETS DEL SISTEMA DE ÓPTICA ACTIVA DE LOS ESPEJOS M1.”

IGNACIO ORTIZ VALDEBENITO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Pedro Toledo Correa
Profesor Correferente: Jose Luis Martí Lara

Diciembre - 20XX

GLOSARIO

Aquí se deben colocar las siglas mencionadas en el trabajo y su explicación, por orden alfabético. Por ejemplo:

VLT: Very Large Telescope.
UT: Unitary Telescope.
AT: Auxiliary Telescope.
VLTI: Very Large Telescope Interferometer.
VISTA: Visible and Infrared Survey Telescope for Astronomy.
VST: VLT Survey Telescope.
ESO: European Southern Observatory.
M1: Espejo Primario

CAPÍTULO 1
DEFINICIÓN DEL PROBLEMA

1.1. DEFINICIÓN

Actualmente, en el VLT del Observatorio Paranal, no existe un procedimiento que permita analizar los registros de log y extraer sets de información bien precisas de estos, bajo ciertas condiciones especificas, con tal de generar batches de data que puedan ser utiles en un futuro.

Esto se debe a que los datos retornados por el sistema de Óptica Activa usan un formato abstracto y poco estructurado. Esto también aplica para los logs retornados durante las operaciones dentro del sistema. Además, estos logs registran todas las actividades y factores presentes durante la operación de los UTs en una noche, generando archivos de gran tamaño con información muy variada.

Estos factores imposibilitan el análisis de la información dispuesta en dichos logs, ya que el costo para reordenar y organizar los datos acorde a lo necesitado supera los beneficios del análisis mismo. A partir del análisis de estos datos, se podría extraer información útil para el diseño de sistemas para la asistencia y mejora de las operaciones de los UTs.

1.2. CONTEXTO

Perteneciente a la ESO, el observatorio Paranal se ubica en el cerro homónimo en el Norte de Chile, y se dedica principalmente a la búsqueda y estudio de galaxias y otras estructuras interestelares. Para lograr estos objetivos, el observatorio cuenta con el VLT, uno de los telescopio ópticos más avanzados del mundo [ESO, 1998].

El VLT debe cumplir con altos estándares tecnológicos, con tal de cumplir la visión de avanzar el entendimiento del Universo mediante la disposición de instalaciones de clase mundial [ESO, 1998].

Como telescopio óptico, el VLT usa una serie de espejos para conducir la luz estelar a instrumentos que capturan la misma en la forma de imagenes, destinada tanto a fines científicos como para la mejora del funcionamiento del telescopio [ESO, 1998].

Para cumplir con estas funciones, el VLT consta principalmente de 4 telescopios primarios, denominados UT, sumado a 4 telescopios auxiliares y otros sistemas complementarios [ESO, 1998].

La forma en la que cada UT capta la luz esta reflejada en el Anexo 1. Primero, la luz es captada

por un espejo principal de 8.2 metros de diametro, denominado M1, el cuál gracias a su forma redirige la luz a un espejo de 0.9 metros de diametro, denominado M2. Luego este último vuelve a redirigir la luz a un tercer espejo eliptico, denominado M3, el cuál redirige la luz a uno de cuatro puntos focales. Desde cada punto focal, la luz es procesada por un sensor de frente de onda Shack-Hartmann, y luego estos datos son procesados por un computador central, el cuál interpreta estos datos como imágenes [ESO, 1998].

Las características físicas del M1 lo vuelven susceptible a sufrir deformaciones por diversos factores, como por ejemplo el efecto de la gravedad cuando se inclina. Estas deformaciones, a su vez, pueden generar aberraciones en las imágenes obtenidas [Wilson *et al.*, 1987].

Para resolver estos problemas, se implementa un sistema en el que el M1 está posicionado sobre 150 propulsores hidráulicos y/o neumáticos, denominado actuador de fuerza. La fuerza que cada actuador ejerce sobre el M1 brinda al mismo de una forma determinada, la cuál permite capturar una imágen de una cierta calidad [ESO, 1998].

Cada imágen tomada es analizada por el sensor Shack-Hartmann, calculando una nueva distribución de fuerzas para los actuadores, la cuál luego es aplicada por los mismos en el M1, brindando a este último de una nueva forma que permita tomar una imágen de mejor calidad [ESO, 1998].

El ciclo anteriormente descrito corresponde al sistema de Óptica Activa, implementada en todos los UTs, el cuál se ejecuta repetidas veces por cada observación hasta lograr una imágen de calida óptima [ESO, 1998].

1.3. ACTORES INVOLUCRADOS

Los actores involucrados en el problema corresponden a los elementos participantes en el sistema de Óptima Activa del telescopio VLT, más especificamente la célula M1, los sensores de frente de onda Shack-Hartmann y el software de control que almacena y analiza los logs y los datos entregados por los sensores [ESO, 2011c].

1.4. OBJETIVOS Y ALCANCE

EL objetivo detrás de la solución busca desarrollar un procedimiento para extraer los datos no estructurados, transformarlos en datos estructurados, y disponerlos para distintos modelos y sistemas que busquen analizar dichos datos con diversos fines, como por ejemplo mantenimiento o técnicas de softcomputing.

Para el caso específico de esta memoría, se tendrá como sistema final de análisis al modelo Neural M1: una red neuronal que es entrenada con datos relacionados al sistema de Óptica

Activa, con el propósito de modelar y predecir la operación de dicho sistema durante una noche.

CAPÍTULO 2
MARCO CONCEPTUAL

2.1. TELESCOPIO VLT

El VLT, sigla para “Very Large Telescope”, es un telescopio ubicado en el Cerro Paranal a 2635 metros de altura, como parte de la “European Southern Observatory” (o conocida por su sigla ESO). Los fines bajo los cuáles se construyó el VLT corresponden a los siguientes [ESO, 1998]:

- El mayor área de colecta posible, según los recursos disponibles.
- La mayor cobertura de longitud de onda, con tal de explotar completamente todas las ventanas atmosféricas.
- Máxima flexibilidad y amplia diversificación instrumental, permitiendo múltiples usos de las instalaciones, incluyendo observaciones simultáneas de múltiples longitudes de onda.
- Capacidad limitada por la difracción de la mayor línea de base posible.
- Optimización de los procedimientos de operaciones científicas, con tal de permitir la explotación, completa y en tiempo real, de la calidad astronómica del sitio y garantizar un máximo retorno científico.

El VLT se compone de una red de 4 telescopios principales idénticos, denominados como “Unitary Telescopes” (o por sus siglas UT), 4 telescopios auxiliares, denominados como “Auxiliary Telescopes” (o por sus siglas AT), el Interferómetro VLT (por sus siglas en inglés VLTI), 1 VISTA y 1 VST.

Cada UT posee un espejo principal (denominado M1) de forma cóncava con 8,2 metros de diámetro, instalado en un montura de 22 metros de largo, 10 metros de ancho y 20 metros de alto. Dicha montura permite al espejo moverse según un eje de coordenadas horizontal, esto es, según azimut (eje horizontal) y altitud (eje vertical) [ESO, 1998].

El M1 está compuesto de espejos más pequeñas, distribuidas en forma de dona. Bajo M1, repartidos en 6 anillos concéntricos, se encuentran 150 actuadores de fuerza axiales; estos son, propulsores hidráulicos y/o neumáticos, donde cada actuador se encuentra debajo de una espejo pequeña respectiva. Estos actuadores dan a M1 una determinada forma óptica, determinada por el patrón de fuerza presentado por los actuadores [ESO, 1998].

El M2 consiste de una espejo de 0.9 metros de diámetro, montada a una distancia de aproximadamente 12.3 metros de M1 a lo largo del eje azimutal, con la espejo de M2 apuntando

hacia M1. El M2 además está montado sobre un mecanismo electromecánico que sujeta y controla su inclinación [ESO, 2011b].

El M3 consiste de una espejo elíptica de 1.24 metros de diámetro mayor con 0.86 metros de diámetro menor. Este se ubica dentro de una torre, posicionada en el orificio central del M1. Este puede rotar alrededor de su eje azimutal [ESO, 2011a].

Los ATs son telescopios con espejos de 1,8 metros de diámetro, encargados de complementar la luz obtenida por los UTs cuando operan bajo el VLTI.

El VLTI corresponde al arreglo de los UTs y ATs funcionando de forma coordinada, recombinando la luz obtenida por cuatro telescopios (los cuales pueden ser UTs y/o ATs) de forma simultánea, cada uno encargado de recombinar secciones específicas del espectro electromagnético. Con este sistema, el diámetro total disponible para captar luz es igual a distancia entre los telescopios [ESO, 1998].

2.2. SISTEMA DE ÓPTICA ACTIVA

La Óptica Activa es un sistema integrado en el M1, encargado de corregir las aberraciones y degradación en la calidad de imágen generadas por las ópticas del espejo [ESO, 1998].

Estas aberraciones suelen ser causadas por la sensibilidad de este a perturbaciones ambientales, cómo las distorsiones térmicas, turbulencia atmosférica, deformación de espejo por ráfagas de viento, errores de manufactura y mantenimiento del telescopio, entre otros. Esta sensibilidad a su vez es causada por la baja proporción entre el grosor y el diámetro del espejo [Wilson *et al.*, 1987].

El ciclo básico del sistema de Óptica Activa se ilustra en la imágen del Anexo 1.

El sensor de frente de onda Shack-Hartmann toma una estrella desde el cielo y la usa como referencia offset, con la cuál monitorea constantemente la calidad óptica de la imágen mediante análisis. Cuando se calcula una desviación considerable de la imágen de su calidad óptima, se descompone dicha desviación en contribuciones ópticas simples (grados de astigmatismo, desenfoque, entre otros.) y posteriormente calcula las correcciones al patrón de fuerza presente en los actuadores de fuerza de M1 [ESO, 1998]. Este nuevo patrón de fuerza cambia la forma de M1, con la cuál se obtiene una nueva calidad de la imágen captada. Luego, con esta imágen, el sensor de frente de onda Shack-Hartmann vuelve a calcular la desviación a la calidad óptima, y si la desviación calculada es considerable, se repite el proceso hasta que se logre la calidad óptima [ESO, 1998] [Wilson *et al.*, 1987].

2.3. SENSOR DE FRENTE DE ONDA

Un sensor de frente de onda es un dispositivo electrónico diseñado para capturar los frentes de onda de una fuente, ya sea en el espectro visible o en infrarrojo. El VLT usa CCD Shack-Hartmann como sensores de frente de onda, con cada UT poseyendo uno en un brazo mecánico específico para los sensores. [ESO, 1998]

El objetivo del uso de sensores Shack-Hartmann es la medición de distorsiones del frente de onda capturado desde la fuente, sobre las cuáles se realiza el proceso de Óptica Activa. [ESO, 1998]

2.4. NEURAL M1

Neural M1 es una red neuronal diseñada para modelar el sistema de Óptica Activa de un UT durante una noche. Este modelado se realiza con la intención de poder predecir el desarrollo de la Óptica Activa en base a un número inicial de ínstancias de corrección de fuerzas.

Para lograr tal cometido, el modelo debe ser entrenado con datos sobre las instancias de corrección de fuerzas durante la Óptica Activa, donde cada instancia va antecedida por una imágen tomada por el sensor de frente de onda y por la distribución de fuerzas en los actuadores, y seguida por otra imágen tomada luego de la corrección y la distribución de fuerzas tras la corrección de las mismas.

2.5. LOGS

Logging se refiere al proceso de registrar diferentes eventos y actividades que ocurren dentro de un sistema de software [Jayathilake, 2011]. Estos registros suelen almacenarse en archivos para su posterior análisis, tanto por parte de desarrolladores como de operadores externos.

Los registros de log son los único tipo de datos que registran, valga la redundancia, información de la operación interna de un sistema de software, por lo que su rol en la industria es importante [Ma y Sun, 2023].

2.6. ANÁLISIS DE LOGS

Originalmente, el análisis de registros de logs era realizado por desarrolladores con el fin de trazar el flujo de ejecución del sistema de software, identificar excepciones y potenciales errores. [Jayathilake, 2011]

Actualmente este enfoque se ha expandido a casos de uso en otros servicios en la industria [Ma y Sun, 2023], debido a que, por la naturaleza de la información contenida en los registros de log, su análisis permite a los operadores detectar, diagnosticar e incluso predecir errores que puedan afectar la disponibilidad y el rendimiento del sistema de software [Jayathilake, 2011].

Ejemplos de estos nuevos casos de uso incluye suplementar información para la detección de intrusiones en redes, recopilación de logs de gran tamaño en forensia digital, entre otros. [Ma y Sun, 2023]

En el pasado, durante la prevalencia del enfoque original, el análisis de registros de log era realizado aplicando revisiones visuales y reglas construidas manualmente. Sin embargo, la complejidad de los sistemas de software actuales ha llevado a la complejización de sus respectivos registros de logs, por lo que ya no es posible depender solamente de los métodos anteriormente mencionados. [Ma y Sun, 2023]

Por esto, en los últimos años se ha desarrollado ampliamente el área del análisis automatizado de registros de logs, mejorando su eficiencia y exactitud mediante la aplicación de tecnologías distribuidas y técnicas de machine learning. [Ma y Sun, 2023]

2.7. ESTRUCTURACIÓN DE LOGS

Los registros de logs generalmente poseen una composición demasiado compleja como para ser interpretada de forma directa y manual; sin el acceso de conocimiento profesional, es difícil seleccionar de forma manual las reglas apropiadas para la comprensión de los registros de log [Ma y Sun, 2023]. Por esta condición es que se refiere a que los registros de log sean “No Estructurados” o “Semi Estructurados”.

Además, el gran tamaño de los archivos de log promedio también se vuelve un problema para el análisis manual de los registros de log [Ma y Sun, 2023].

Debido a esto, durante los últimos años, se han desarrollado herramientas, procedimientos y frameworks para el análisis automático de registros de log, y una parte considerable de los esfuerzos realizados se enfocan en la “Estructuración” de los registros de log.

Actualmente, el workflow de análisis automático de registros de log se divide en 2 etapas centrales [Ma y Sun, 2023]:

- Log Parsing: Se toman los registros semi-estructurados de log y se generan plantillas a partir de estos. Una plantilla es una sentencia estructurada que se repite entre varios registros de log, dividiéndose en tokens estáticos y valores dinámicos. [Ma y Sun, 2023]
- Feature Extraction: Se aplican las plantillas generadas sobre los registros de log para

obtener las características, esto es, las variables dinámicas, de los mismos. [Ma y Sun, 2023]

CAPÍTULO 3
PROPUESTA DE SOLUCIÓN

3.1. Identificación de datos deseados

Los registros de log proporcionan una gran cantidad de información sobre las operaciones del VLT en una noche. Sin embargo, Neural M1 no necesita ingerir toda esta información; acorde a su objetivo, la red neuronal solo necesita aquella información que le permita predecir el proceso de Óptica Activa. Debido a esto, antes del diseño del procedimiento ETL, es necesario hacer un estudio de los registros de log y determinar aquellos que sean de utilidad para Neural M1.

3.1.1. Datos principales

Durante el proceso de Óptica Activa, las correcciones de las distribuciones de fuerza en los actuadores son realizadas en base imágenes tomadas previamente por los sensores. Se debe considerar, además, que el proceso es iterativo, por lo que después de una corrección, se toma una nueva imágen con la nueva distribución de fuerzas para una nueva corrección [ESO, 1998].

Con estas consideraciones en mente, se puede inferir para una instancia de corrección, debe existir una imágen tomada antes y una imágen tomada después, donde esta última correspondería a la imágen previa de la siguiente instancia de corrección.

Además, para los fines de Neural M1, se hace necesario conocer los valores de las distribuciones de fuerza de los actuadores resultantes, ya que esta es la cantidad optimizada durante la Óptica Activa [ESO, 1998].

3.1.2. Datos adicionales

Sumado a los datos anteriormente descritos, existen otras características que son de interés, debido a que complementan y/o influyen en el comportamiento de la Óptica Activa.

Dentro de estas características se encuentran:

- Cantidades del torque, tanto en altitud como en acimut, aplicado al motor durante el tracking del telescopio [ESO, 1998].

- Tantos en ángulos como la posición actuales del UT, considerando altitud y acimut [ESO, 1998].
- Aberraciones de imágen, obtenido después de una corrección, con tal de evaluar el rendimiento deseado [Wilson *et al.*, 1987].

3.2. Almacenamiento de datos

Una vez identificados los datos a extraer desde las líneas de log, se hace necesario plantear como se almacenaran los mismos despues de obtenerlos.

Tener esta claridad previo al desarrollo del procedimiento ETL permite tener la claridad de que transformaciones realizar a los datos extraídos.

3.2.1. Estructura de los datos

Según lo visto anteriormente, los datos principales consisten en imágenes, distribuciones de fuerzas, etc. asociados a instancias de corrección de la forma del espejo M1. Más específicamente, se tiene que cada instancia de corrección cambia, generalmente, los valores de estos datos.

Se puede considerar entonces agrupar los datos principales según la instancia de corrección asociada. Por lo señalado anteriormente, cada instancia de corrección tendría asociado un set de datos previos a la corrección y otro set de datos similar luego de la misma. Entonces, cada corrección correspondería a una tabla con ambos sets de datos como atributos de la misma.

Esto puede llevar a un problema, ya que en el escenario de dos correcciones consecutivas, el set de datos post-corrección de la primera instancia y el set de datos pre-corrección de la segunda instancia serían los mismos, por lo que estos datos aparecen repetidos en ambas tablas.

Para solucionar este problema, una vez agrupados los datos, se opta por separar las correcciones y los sets de datos asociados en tablas distintas pero relacionadas entre sí con llaves foráneas. Al mantener entidades distintas con múltiples relaciones por tabla, se logra normalizar la información.

Siguiendo con el caso anterior, se tendrían dos tablas para las correcciones más tres tablas para los sets de datos: una tabla para el set de datos pre-corrección asociada a la primera corrección, una tabla para el set de datos post-corrección asociada a la segunda corrección y una tercera tabla para el set de datos entre ambas correcciones, a las cuales se relaciona usando llaves foráneas desde las tablas de las correcciones.

Finalmente, considerando que el set de datos se compone de datos de naturaleza variada (rutas a imágenes, valores numéricos de fuerzas, etc), se decide agrupar los datos en tablas propias según su naturaleza.

Recapitulando, con respecto a los datos asociados a correcciones, se opta almacenar los datos en un modelo de tablas relacionales, compuesta de una tabla central dedicada a las instancias de corrección y varias tablas para los datos correspondientes al antes y después de una corrección.

— Imagen de las tablas de correction —

Con respecto a los datos adicionales, debido a que estos no cambian con las correcciones, se decide almacenar estos en una tabla propia, conteniendo los atributos clave y sin necesidad de relaciones con otras tablas.

— Imagen de tabla de additional-data —

3.2.2. Formato de almacenamiento

Después de diseñar la estructura de los datos, se hace necesario saber qué tecnologías usar para llevarla a cabo.

Los modelos dejan ver que una base de datos SQL sería la mejor opción. Si bien esta puede permanecer como una opción para futuros usos de los datos extraídos, esta opción no sería óptima para el objetivo de esta memoria, debido a que el procedimiento ETL a desarrollar se acoplara a nivel de código a Neural M1, por lo que subir los datos a un servidor SQL para luego bajarlos en el mismo sistema es redundante e innecesario.

Por ende, y considerando el lenguaje de programación usado, la mejor opción es la librería Pandas, más específicamente la estructura Dataframe, la cuál a nivel técnico es homóloga a una tabla SQL dentro de un ambiente Python.

3.2.3. Tecnología de almacenamiento

Después de diseñar la estructura de los datos, se hace necesario saber qué tecnologías usar para llevarla a cabo.

Los modelos dejan ver que una base de datos SQL sería la mejor opción. Si bien esta puede permanecer como una opción para futuros usos de los datos extraídos, esta opción no sería óptima para el objetivo de esta memoria, debido a que el procedimiento ETL a desarrollar se acoplara a nivel de código a Neural M1, por lo que subir los datos a un servidor SQL para luego bajarlos en el mismo sistema es redundante e innecesario.

Por ende, y considerando el lenguaje de programación usado, la mejor opción es la librería Pandas, más específicamente la estructura Dataframe, la cuál a nivel técnico es homóloga a una tabla SQL dentro de un ambiente Python.

3.3. Preprocesamiento de lineas

Los registros de log poseen cientos de miles de líneas, por lo que un proceso de filtrado aplicado directamente sobre estos sería costoso en tiempo y poder de cómputo.

Por ende, es prudente procesar los registros de log de forma previa, aplicando transformaciones que retornen un conjunto de líneas más apto para el filtrado.

3.3.1. Remoción de tokens estáticos comunes

Tras revisar los registros de log, se puede notar que existe un patrón recurrente en todas las líneas:

— Ejemplo de líneas del log —

Estos tokens iniciales aportan información que es redundante dentro de la misma línea y, a su vez, que es estática, ya que la misma no cambia entre las distintas líneas. De este modo, se puede concluir que dicha información no es necesaria para el objetivo de esta memoria.

Por lo mismo, con tal de facilitar la extracción de datos, se procede a eliminar estos tokens estáticos.

3.3.2. Cruce con archivo de observaciones

Parte del objetivo de esta memoria es garantizar que la información extraída sea fidedigna y correcta. Por ende, es necesario asegurar que las observaciones a tratar sean exitosas.

Además, no todas las observaciones son usadas en el proceso de Óptica Activa; las observaciones de calibración no son consideradas para este proceso, solo las de adquisición y científicas.

Los registros de log no muestran de qué tipo es cada observación, por ende es necesario extraer esta información de un documento externo y cruzarla con las líneas de observaciones ya extraídas, con tal de así poder obtener las líneas que sí influyen en la Óptica Activa.

1. ESO raw

La ESO posee una página web donde se puede acceder de forma libre a información sobre las observaciones realizadas por el VLT. Dentro de esta información, se puede extraer las observaciones realizadas, la hora en que se realizaron y el tipo de observación, entre otros atributos.

Para obtener el documento de observación necesario para el procedimiento ETL, se deben añadir los siguientes campos:

— Imagen de la pagina ESO raw con los campos puestos —

Estos campos permiten obtener todas las observaciones cientificas y de adquisición de todos los instrumentos en una noche determinada.

Es posible realizar una petición POST con los campos deseados a la página, y esta luego retorna un archivo csv con las observaciones de la noche descrita para todos los instrumentos disponibles durante la misma.

2. Segmentación de los periodos de observación

   Una vez cargado el documento, se procede a usar el mismo para delimitar los horarios de inicio y término de cada periodo de observación valido, para luego usar estos límites en el filtrado de las observaciones obtenidas desde el registro de logs.

   Inicialmente, se pueden crear estos ”bloques de tiempoüsando la hora de inicio de cada observación junto con el tiempo de exposición de la misma. Es importante que, al momento de crear estos bloques, se mantenga la distinción de bajo que instrumento se genera la observación en cuestión.

   Luego de este paso, es natural intuir que algunos de estos bloques para un mismo instrumento se puedan sobrelapar, lo cuál provocaría que se duplicaran algunas líneas de log tras filtrar. Por ende, tras crear los bloques de tiempo, se debe cerciorar que aquellos bloques que se sobrelapen en horario para un mismo instrumento sean unidos.

   Sumado a esto, se conoce que los telescopios estarán en funcionamiento por ciertos intervalos de tiempo previo y posterior a una observación [ESO, 1998]. Debido a estos se considera añadir tres margenes paramétricos de tiempo:

   - Margen inferior: Cantidad de segundos previo a un bloque de tiempo.
   - Margen superior: Cantidad de segundos posterior a un bloque de tiempo.
   - Umbral intermedio: Cantidad de segundos entre bloques de tiempo para un mismo instrumento (con margenes incluidos) donde, si la distancia horaria es menor al umbral, se procede a unir dichos bloques.

   Si bien se conoce que cada observatorio es operativo antes y después de una observación formal, la cantidad de tiempo específica (X segundos antes y Z segundos después de cada observación) es desconocida, y no se asegura que sean cantidades estables para cada UT o para una noche específica.

   Es por esta razón, que los margenes y el umbral son paramétricos, con tal de buscar la cantidades bajo las cuales se puede obtener mejores resultados.

3. Clasificación según cruce

   Con los bloques de tiempo formados, solo resta filtrar las líneas de log en base de las mismas.

   Para este proceso, solo basta con incluir todas las líneas de log cuyo horario se encuentre dentro de los bloques de tiempo.

3.4. Extracción de datos

Para extraer los datos, el método tradicional es el parsing usando plantillas de líneas de log. El objetivo es extraer los datos deseados de las líneas de texto, y disponerlos en los formatos adecuados.

Debido a la proliferación y estandarización de este método, se procederá a usar el mismo para las líneas previamente procesadas.

3.4.1. Plantillas de lineas

Las plantillas corresponden a una secuencia de carácteres que especifican los valores deseados dentro de una línea de texto.

Estas pueden ser creadas manualmente o generadas automáticamente. Debido a que se conoce la cantidad específica de datos requeridos, se prefiere no complejizar el procedimiento con un sistema de creación de plantillas automático, y en cambio crear las plantillas manualmente.

Estas plantillas se disponen finalmente al procedimiento ELT, el cuál aplicará parsing con las plantillas de forma automática.

El formato final de las plantillas varía según el método de parsing usado, por lo que se muestran imágenes de las mismas en las subsecciones correspondientes.

Sin embargo, según el análisis realizado en la sección 3.X, se puede estimar que se necesitan unas 16 plantillas para obtener los datos deseados; una plantilla para cada línea a parsear.

3.4.2. Parsing

El parsing de líneas de log se refiere al análisis de las líneas de texto presentes en los registros de log con el fin de identificar los tokens presentes y detectar los datos relevantes con el fin de extraerlos [Jayathilake, 2011].

Existen varias formas de realizar parsing en registros de log; escoger la adecuada no es trivial, ya que se presentan dos desafíos principales a la hora de aplicar parsing en logs [Jayathilake, 2011]:

Las arquitecturas de software grandes y complejas generan grandes cantidades de información de log mientras se ejecutan, por lo que la forma tradicional de construir manualmente expresiones regulares es demasiado costosa. Las funciones de sistemas de software complejos y actualizaciones de negocio de alta frecuencia llevan a actualizaciones más frecuentes de plantillas de línea, incrementado así en gran medida la diversidad de plantillas de línea.

La problemática 1 se orienta principalmente al parsing general de registros de log, el cuál no es el caso para este trabajo; ya que se tiene claridad de los datos requeridos y de las líneas específicas que los poseen, el uso de expresiones regulares y plantillas creadas manualmente no es descartado.

La problemática 2, si bien también se orienta hacia el parsing general de registros de log y a la creación automática de plantillas de log, se puede extender su cuestionamiento a la creación de plantillas manuales; a medida que se actualicen las reglas de negocio de la Óptica Activa o las necesidades de Neural M1, las plantillas manuales creadas pueden aumentar.

Como se mencionó en el Capítulo 1, el sistema de logging de Óptica Activa tiene décadas de servicio continuo, por lo que es poco probable que se realicen cambios en esta área. Por el lado de Neural M1, es posible que puedan existir cambios en los datos requeridos en un futuro, sin embargo el análisis realizado en la sección 3.X, se puede asegurar que los cambios se mantendrían dentro de ese modelo de datos. Por ende, cualquier cambio futuro no incrementa el número de plantillas en gran medida.

Debido a todo lo anterior, se decide optar por un modelo sencillo de parsing, donde se itera sobre las secciones de log obtenidas tras el preprocesamiento, y dentro de cada iteración se aplica de forma greedy cada plantilla sobre la línea seleccionada hasta que una calce. En caso de calzar, se extraen los datos acordes a la plantilla y se almacenan estos en una lista.

Para aplicar el parsing en las líneas de texto, existen varios métodos y librerías diseñadas para la extracción de datos desde strings. Para este trabajo, se analizaron dos opciones posibles y se compararon según eficiencia y exactitud.

1. Parsing con Expresiones Regulares

   En este caso, se tienen las plantillas en forma de expresiones regulares, donde los tokens deseados estan marcados como grupos dentro de las mismas plantillas.

   Para este caso, se iteran en todas las líneas de log filtradas por observación, y se detectan las líneas que contienen los datos deseados usando ciertos tokens claves detectados al estudiar el formato del archivo de log.

   Luego de detectar las líneas, se aplican a estas todas las plantillas hasta que se pueda parsear correctamente con una.

2. Parsing con Template Text Parser

   Para este caso, se usa la librería externa Template Text Parser, o TTP, la cuál, de forma similar al caso anterior, permite escribir los tokens deseados en la misma plantilla, y luego durante la extracción solo retorna los datos llamados anteriormente.

   Como es una librería diseñada para estos fines exclusivos, se procede solo a ejecutar un método de la misma para parsear.

Para ambos casos, los datos parseados quedan en formato json, donde la llave corresponde al nombre de la caracteristica (definida desde la plantilla con TTP y definida desde el codigo con Regex) y el valor corresponde al dato parseado.

Se verificó con ambos métodos, y si bien ambos funcionan, eventualmente se decanto por el primer método. Esto más que nada llevado por una necesidad de optimización, ya que si bien TTP es una libreria diseñada especificamente para este tipo de tareas, toma una cantidad de tiempo considerablemente mayor en parsear comparado con el método de expresiones regulares.

Independiente del origen del dato, todos poseen campos de nombre ”groupz la mayoría posee también un campo ”label”. Se usará el campo ”group”para definir el tipo del dato (distribución de fuerza, imágen, etc.) y el campo ”label”para identificar subdivisiones dentro de cada tipo de dato cuando sea conveniente.

Estos campos se tornan particularmente útiles, cuando se considera que la información para un dato puede estar repartida en varias líneas, como es por ejemplo el caso de las distribuciones de fuerza, donde las fuerzas de los 150 actuadores están siempre repartidos en 6 líneas de 25 actuadores, y que a su vez todas esas líneas están separadas de la ID de la distribucón, que se encuentra en su línea propia.

— Ejemplo de linea de log de fuerzas —

De esta forma, mediante el uso de estos campos, se puede dejar registro de la información de un mismo dato para su posterior unión.

3.5. Guardado de los datos

Una vez parseados los datos, se deben crear y validar los dataframes donde se almacenaran estos mismos, siguiendo la filosofía descrita en la sección 3.2.

Es necesario, como se mencionó, no solo crear los dataframes de forma coherente, sino que también validar que los datos ingresados sean correctos ya sea en formato, en contexto, etc.

3.5.1. Construcción de los dataframes

En la sección 3.2.1. se definieron las tablas a desarrollar con sus relaciones y atributos correspondientes, por ende, los dataframes a crear deben seguir la misma estructura.

Asi mismo, se crearan 4 dataframes:

- Uno para las instancias de corrección, donde para cada instancia se guardan las referencias a las imágenes y distribuciones de fuerza anteriores y posteriores.
- Uno para las distribuciones de fuerza, donde para cada distribución se guardan los pesos de los 150 actuadores y el timestamp del registro en el log.
- Uno para las imágenes tomadas, donde para cada imágen se guardan los tiempos de inicio y término de lectura, el tiempo de inicio de exposición, el tiempo de integración, el sensor que leyo la imágen y la dirección local al archivo .fits correspondiente.
- Uno para todos los datos adicionales, donde para cada dato se guarda el grupo (relacionado con la cantidad con la que trata, ya sea altitud, acimut, imágen, etc.), la etiqueta (relacionado con la cualidad específica que maneja esa cantidad, sean aberraciones, torque, etc.), el dato numérico o en texto. Para lo último, se decide guardar el dato en formato entero, flotante y texto, con fines de respaldo, sumado a un campo de tipo de dato que describe cuál es el formato correcto.

De esta forma, los dataframes finales poseen los siguiente esquemas finales:

— Imagen de los schemas de dataframes —

3.5.2. Ingesta a los dataframes

Debido a que los datos parseados se encuentran en formato json, corresponde iterar por todos los datos, y migrar los datos desde el formato json a dataframe.

En la sección 3.4.2. se mencionaron las llaves ”groupz ”label”para diferenciar los tipos de datos. Estas llaves se usan en este proceso, con tal de poder identificar el tipo de dato, y según aquello, migrar los datos de forma acorde.

Como ejemplo, se conoce que los datos de distribución de fuerzas posee un campo con las fuerzas de los actuadores, y que ningún otro tipo de dato posee tal campo. Entonces, al tener registro del tipo de dato en la llave ”group”, se puede validar que datos poseen ese campo y extraerlos sin temor a falla.

Asi mismo, se puede concatenar la información de ciertos datos que se encontraban en múltiples líneas. Tras revisar el campo ”group”, se revisa que no existan datos en el dataframe

con información faltante, y se revisa el campo ”label”para verificar que tipo de información específica es la que falta.

3.5.3. Validación de los dataframes

Eventualmente durante el proceso de ingesta a los dataframes se producirán errores, principalmente por errores en los datos desde el origen o por las convenciones tomadas para el desarrollo de los filtros.

Debido a ello, luego de la ingesta, se deben validar los datos dentro de los dataframes con tal de asegurar que dicha información es correcta y de útilidad.

Para cada tipo de dato, el proceso de validación es distinto:

- Para las instancias de corrección, se deben remover las instancias con ID nulo, y también se remueven aquellas instancias que no poseen distribución de fuerza posterior a la corrección.
- Para las distribuciones de fuerza, se deben remover las instancias con ID nulo y también se remueven aquellas distribuciones que no son antecedidas y/o seguidas por imágenes únicas.
- Para las imágenes, solo se remueven las instancias con ID nulo.

El dataframe de additional-data no es validado, debido a ser datos de carácter miscelaneo y secundario.

3.5.4. Relación de los dataframes

Finalmente, una vez se valida la información presente en los dataframes, se procede a crear las relaciones mencionadas en la sección 3.2.2.

Para esto, simplemente se hacen calzar los timestamp de los datos de imágen y de distribución de fuerza, y luego se entrega sus ID a la instancia de corrección para incluirlas en sus campos de llaves foráneas.

CAPÍTULO 4
VALIDACIÓN DE LA SOLUCIÓN

Se debe validar la solución propuesta. Esto significa probar o demostrar que la solución propuesta es válida para el entorno donde fue planteada.

Tradicionalmente es una etapa crítica, pues debe comprobarse por algún medio que vuestra propuesta es básicamente válida. En el caso de un desarrollo de software es la construcción y sus pruebas; en el caso de propuestas de modelos, guías o metodologías podrían ser desde la aplicación a un caso real hasta encuestas o entrevistas con especialistas; en el caso de mejoras de procesos u optimizaciones, podría ser comparar la situación actual (previa a la memoria) con la situación final (cuando la memoria está ya implementada) en base a un conjunto cuantitativo de indicadores o criterios.

4.1. EJEMPLO DE COMO CITAR TABLAS

Se colocó una tabla que se puede referenciar también desde el texto

4.2. Algoritmo

El código del modelo fue desarrollado con Python 3.10.8, siguiendo el modelo descrito durante el Capítulo 3. Se puede encontrar el mismo en un repositorio público en Github [1].

4.2.1. Implementación

El flujo del algoritmo final es puede notar en el Anexo 2, donde cada bloque rectangular corresponde a una función, mientras que cada bloque con borde inferior irregular corresponde a un archivo, ya sea de texto plano o dataframe.

El algoritmo se divide en los siguientes pasos:

1. log_formatting: Cumple con el proceso planteado en la sección 3.3.1, donde se remueven los tokens de fecha y hora iniciales, debido a su redundancia a nivel de línea, además de corregir otros detalles de formato con tal de facilitar el posterior análisis.

[1] link al repo github

2. obs_filtering: Cumple con el proceso descrito en la sección 3.3.2, donde se obtiene un archivo csv con las observaciones de la noche seleccionada, para luego formar bloques de tiempo basados en los tiempos que tarda cada observaciones, más el efecto de parámetros de margen y unión, donde finalmente se filtran las líneas de log con tal de quedar solo con aquellas cuya fecha se encuentre dentro de alguno de esos bloques de tiempo.

3. log_parsing_regex: Cumple con el proceso descrito en la sección 3.4, donde a base de una archivo plano de plantillas, se itera sobre las líneas de log, aplicando cada plantilla por línea hasta encontrar alguna que calze y guardar la data extraída en un json final.

   Como se mencionó en la sección 3.4.2, el método para realizar el parsing es mediante Expresiones Regulares, lo cuál se manifiesta tanto en el archivo de plantillas que se ingresa a la función, junto con las líneas a parsear, como en el método de parsing que se aplica a línea de log.

4. generate_dataframes: Cumple con el proceso de la sección 3.5.2, donde se menciona el traslado de los datos del json generado en la función anterior a un conjunto de dataframes, usando los campos "groupz "labelçomo etiquetas para reconocer a qué dataframe destinar los datos.

   De esta función se generan 4 dataframes, como se señalo en la sección 3.5.1: df_corrections, para las instancias de corrección; df_images, para las imágenes capturadas; df_f_dist, para las distribuciones de fuerza de los actuadores; y df_additional_data, para los datos adicionales relacionados con la Óptica Activa.

5. validate_forces: Cumple con el segundo item de la sección 3.5.3, donde se especifica la validación de los datos dentro de df_f_dist generado en la función anterior.

   Principalmente, se eliminan los datos con ID nulo, además de remover aquellos que no sean antecedidas y/o seguidas por imágenes únicas.

6. validate_images: Cumple con el tercer item de la sección 3.5.3, donde se especifica la validación de los datos dentro de df_images generado en la función anterior.

   Solamente se eliminan aquellos datos con ID nulo.

7. link_images: Cumple con el requisito de la sección 3.5.1, sobre df_images manteniendo la dirección local del archivo .fits correspondiente como un atributo de los datos del dataframe.

   En esta función, se ingresa a la carpeta donde estan almacenados los archivos .fits, se extrae el número de ID y el tiempo de exposición del header de cada .fits y se usan estos para buscar el dato de df_images que le sea correspondiente. En caso de encontrarlo, se guarda su dirección local en el atributo correspondiente del dato.

8. validate_corrections: Cumple con el primer item de la sección 3.5.3, donde se especifica la validación de los datos dentro de df_corrections generado en la función anterior.

   Principalmente se eliminan los datos con ID nulo, además de remover aquellos sin distribución de fuerza posterior a la corrección.

9. link_dataframes: Cumple con el proceso descrito en la sección 3.5.4, en la que se indica como calzar los datos de un dataframe con los datos de dataframe df_corrections mediante los timestamp de estos datos, comparando estos para luego copiar la ID del dato de la cantidad deseada en un campo designado dentro del dato correspondient en df_corrections.

   Para esta función, como la construcción de relaciones entre df_images con df_corrections y entre df_f_dist con df_corrections es identico, por lo que se pudo generalizar y paramétrizar esta construcción en una misma función, que varía con el dataframe de la cualidad deseada, el nombre de la cualidad deseada y el campo de tiempo dentro del dataframe, como parámetros de entrada.

4.2.2. Ejecución

Para ejecutar el código desarrollado, se debe entregar la fecha y el número del UT, en ese orden, del registro a analizar. Para el apropiado funcionamiento del algoritmo, el archivo de log debe estar previamente cargado en el sistema local.

Además se desarrolló un sistema de flags opcionales que otorga la posibilidad de habilitar o deshabilitar las funciones descritas en la sección 4.2.1.

– Imagen de las flags del código –

Existen 2 posibles modos de retorno de los dataframes generados: por consola o por csv. Estos se habilitan con las flags -c y -s, respectivamente.

4.3. Pruebas

Para corroborar el correcto funcionamiento del algoritmo, es necesario analizar los resultados del mismo, junto con revisar el rendimiento y exactitud de la ejecución.

4.3.1. Metodología

Para poder analizar el desempeño y resultados del algoritmo, primero se realiza un conteo manual de las instancias de corrección, las distribuciones de fuerza y las imagenes presenten en una muestra de archivos de log. Luego, se ejecuta el algoritmo para procesar los mismos archivos de log, y se comparan los resultados.

La muestra se compone de los siguientes archivos de log:

- wt1tcs.2025-08-03.log : Archivo de log para la observación del UT 1 durante la noche del 4 de Agosto del 2025.
- wt1tcs.2025-08-04.log : Archivo de log para la observación del UT 1 durante la noche del 5 de Agosto del 2025.
- wt3tcs.2025-08-03.log : Archivo de log para la observación del UT 3 durante la noche del 4 de Agosto del 2025.
- wt3tcs.2025-08-04.log : Archivo de log para la observación del UT 3 durante la noche del 5 de Agosto del 2025.
- wt4tcs.2025-08-03.log : Archivo de log para la observación del UT 4 durante la noche del 4 de Agosto del 2025.
- wt4tcs.2025-08-04.log : Archivo de log para la observación del UT 4 durante la noche del 5 de Agosto del 2025.

Una vez realizadas las ejecuciones, se procede a analizar las estadísticas de los resultados, más específicamente el número de instancias de corrección, distribución de fuerzas e imágenes obtenidas, además del tiempo de ejecución y el uso de memoria durante la ejecución.

4.3.2. Resultados

Los resultados encontrados de forma manual para cada archivo de log se refleja en la Tabla 1:

Tabla 1: Cantidades encontradas de forma manual

Nombre de archivo de log	Número de líneas	Número de instancias de corrección	Número de distribuciones de fuerza	Número de imágenes tomadas
wt1tcs.2025-08-03.log	427195	626	586	851
wt1tcs.2025-08-04.log	384729	598	562	802
wt3tcs.2025-08-03.log	409830	537	277	908
wt3tcs.2025-08-04.log	371450	546	262	741
wt4tcs.2025-08-03.log	444126	491	287	786
wt4tcs.2025-08-04.log	422312	495	358	713

Luego, las cantidades encontradas en los mismos archivos por el algoritmo se encuentran en la Tabla 2:

Tabla 2: Cantidades encontradas por el algoritmo

Nombre de archivo de log	Número de líneas	Número de instancias de corrección	Número de distribuciones de fuerza	Número de imágenes tomadas	Tiempo de ejecución (en segundos)
wt1tcs.2025-08-03.log	427195	4	5	11	61.92
wt1tcs.2025-08-04.log	384729	31	33	54	89.33
wt3tcs.2025-08-03.log	409830	4	5	8	69.49
wt3tcs.2025-08-04.log	371450	19	19	46	79.14
wt4tcs.2025-08-03.log	444126	3	3	4	59.18
wt4tcs.2025-08-04.log	422312	18	18	31	103.25

Es evidente la amplia diferencia entre ambas cantidades, sin embargo, existen motivos para explicarlo.

Primero, se debe mencionar que la extracción manual se realizó mediante un simple conteo de lineas de log según ciertos tokens clave, mientras que para la extracción con el algoritmo se aplicó lógica durante las validaciones y filtros que permiten un conteo con información más certera, aunque reducida en tamaño.

Ejemplos de reglas incumplidas en los logs: se ha visto que en los registros de log, existen varias fotos que se pueden tomar consecutivamente en lapsos de unos pocos segundos; distribuciones de fuerza que no son anticipados ni seguidos directamente por una toma de imágen; entre otros.

– Foto de ejemplo: varias fotos seguidas desde el el log–

Estos casos, son detectados por el algoritmo, evitando el parseo de estos datos extra.

El otro motivo de porque la diferencia en el número de cantidades es la filtración por observación: la cantidad de obesrvaciones totales es baja, lo cuál inmediatamente deja fuera a una buena parte de las cantidades presentes en el archivo de log, como se puede ilustrar en la siguiente imágen:

– Foto del log del nm1e, mostrando todas las lineas eliminadas –

En la imágen anterior, los nombres mostrados equivalen a líneas de imágen, desechadas del

parsing final, debido a no calzar con algún horario de observación.

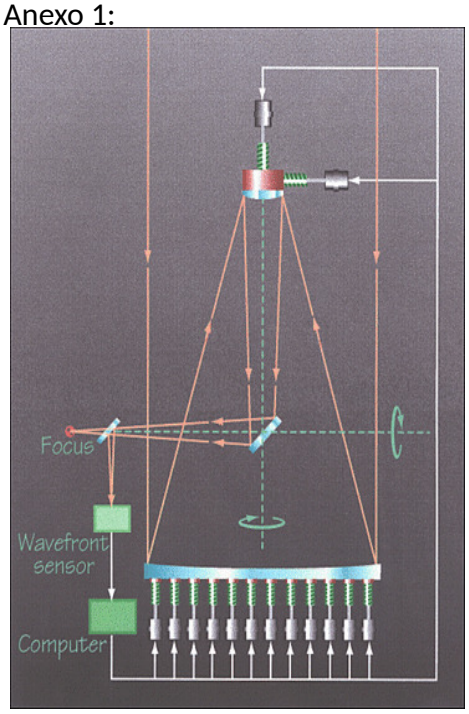
Ahora bien, como se menciono en la sección 3.3.2, el filtro de observaciones se realiza con tres parámetros, los cuáles son de asignación directa dentro del código. Por ende, la modificación de estos parámetros ampliaría el rango de búsqueda de las cantidades en el archivo de log. Actualmente, se definieron los margenes anterior y posterior como 10 segundos cada uno, y el umbral de unidad de los bloques de observación se encuentra en 30 segundos.

Es posible aumentar estos paramétros, sin embargo tampoco se pueden llegar a extremos muy altos que terminen volviendo obsoleto al proceso de filtro por observación.

Se deja abierta la posibilidad de un estudio futuro con el cuál analizar los valores óptimos para los parámetros.

ANEXOS

- Anexo 1:



REFERENCIAS BIBLIOGRÁFICAS

[ESO, 1998] ESO (1998). *The VLT White Book*, volumen 1. ESO.

[ESO, 2011a] ESO (2011a). The m1 cell and m3 tower. `https://www.eso.org/sci/facilities/paranal/telescopes/ut/m1cellm3.html`.

[ESO, 2011b] ESO (2011b). The secondary mirror, m2 unit. `https://www.eso.org/sci/facilities/paranal/telescopes/ut/m2unit.html`.

[ESO, 2011c] ESO (2011c). The vlt active optics system. `https://www.eso.org/sci/facilities/paranal/telescopes/ut/actopt.html`.

[Jayathilake, 2011] Jayathilake, P. W. D. C. (2011). A novel mind map based approach for log data extraction. (6).

[Ma y Sun, 2023] Ma, J., L. Y. W. H. y Sun, G. (2023). Automatic parsing and utilization of system log features inlog analysis: A survey. (21).

[Wilson *et al.*, 1987] Wilson, R. N., Franza, F., y Noethe, L. (1987). Active optics i. a system for optimizing the optical quality and reducing the costs of large telescopes. 34(4):485–509.