UNIVERSIDAD DE LA FRONTERA
FACULTAD DE INGENIERÍA Y CIENCIAS
DEPTO. CS. COMPUTACIÓN E INFORMÁTICA

"MINERÍA DE DATOS PARA LOGS DE ALMA COMMON SOFTWARE"

ANTEPROYECTO DE TRABAJO DE TITULACIÓN
PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL MENCIÓN
INFORMÁTICA

Profesor Guía: DR. PATRICIO ABEL GALEAS ECHEVERRÍA
Profesor Co-Guía: DRA. GLORIA MILLARAY JULIA CURILEM SALDIAS

JAVIER IGNACIO FUENTES MUÑOZ
2015

ANTEPROYECTO DE TRABAJO DE TITULACIÓN

TÍTULO: MINERÍA DE DATOS PARA LOGS DE ALMA COMMON SOFTWARE

NOMBRE: JAVIER IGNACIO FUENTES MUÑOZ

Nº DE MATRÍCULA: 17627072110

CARRERA: ING. CIVIL INDUSTRIAL MENCIÓN INFORMÁTICA

NIVEL CURSADO: XII

JAVIER IGNACIO FUENTES MUÑOZ
Alumno

DR. PATRICIO ABEL GALEAS ECHEVERRÍA
Profesor Guía

DRA. GLORIA MILLARAY JULIA CURILEM SALDIAS
Profesor Co-Guía

MG. NATACHA ALEJANDRA PINO ACUÑA
Directora de Carrera

DR. CARLOS FERNANDO CARES GALLARDO
Director del Departamento

Índice

1. Acerca de ALMA **1**
- 1.1. Problemática General . . . 1
- 1.2. Objetivo General . . . 2
- 1.3. Objetivos Específicos . . . 2

2. Antecedentes Generales **3**
- 2.1. ALMA Common Software . . . 3
- 2.2. Logs . . . 3
  - 2.2.1. Definición de Log . . . 3
- 2.3. Pre-procesamiento de datos . . . 4
  - 2.3.1. Documentos legibles por máquinas . . . 4
  - 2.3.2. Tratamientos de texto para los eventos . . . 4
- 2.4. Procesamiento de datos . . . 5
  - 2.4.1. SVM . . . 5

3. Materiales y Métodos **7**
- 3.1. Metodología . . . 7

4. Programa de Trabajo **8**
- 4.1. Descripción de las actividades . . . 8
- 4.2. Carta Gantt . . . 8

5. Nomenclatura **9**

6. Bibliografía **10**

Índice de figuras

1. Hiperplano lineal de separación de dos clases: (**a**) Margen máximo (**b**) Margen blando. Los datos destacados con círculos, son vectores de soporte . . . 5
2. Efecto de la asignación del espacio de entrada en un espacio de características dimensional superior, donde es posible un plano de separación lineal. . . . 6
3. Proceso KDD . . . 7
4. Carta Gantt . . . 8

Listings

1. Ejemplo un evento XML de log . . . 3

1. Acerca de ALMA

El proyecto ALMA (Atacama Large Millimeter/submillimeter Array) es un trabajo colaborativo entre el European Southern Observatory (ESO), el National Radio Astronomy Observatory (NRAO) y el National Astronomical Observatory of Japan (NAOJ). Este gran proyecto astronómico consiste en un arreglo de 66 antenas ubicado en el desierto de Atacama, en el norte de Chile. ALMA tiene como propósito observar el cielo, en búsqueda de nuestros orígenes cósmicos. Esta observación es guiada por científicos y astrónomos del mundo, quienes solicitan espacios de observación. ALMA Common Software (ACS) provee una plataforma para todos los sistemas del observatorio, la cual se basa en un modelo de componentes de software distribuido. Asimismo, ACS genera un archivo de registros (log) con eventos que dan cuenta del funcionamiento del sistema, tales como: errores, alertas, información de ejecución de programas, interrupciones del sistema, etc.[1].

1.1. Problemática General

ACS es un software distribuido altamente complejo. Como toda infraestructura de software, este presenta fallas en su funcionamiento las que pueden llegar a provocar la detención completa del sistema, provocando retrasos en las tareas de observación. Gran parte del funcionamiento de ACS es permanentemente registrado en archivos de logs, los cuales suman diariamente grandes cantidades de información sobre la operación y eventos asociados a diferentes niveles de alerta. Sin embargo, dada la gran cantidad de registros generados por este sistema, resulta complejo el poder analizar estos datos en forma eficiente y efectiva.

Alma tiene un infraestructura altamente compleja, lo que se refleja en sus sistema de generación de logs, por lo que es natural encontrar problemáticas asociadas a numerosas causas y por este motivo es necesario buscar y analizar estas posibles problemáticas. El problema se plantea complejo tanto por la diversidad de los logs como por su cantidad.

Por otra parte, se ha desarrollado herramientas y metodologías de minería de datos que tienen por objetivo el apoyar el análisis de grandes volúmenes de datos de manera de poder extraer conocimientos nuevos y útiles. Estas herramientas son capaces de procesar grandes volúmenes de información y encontrar relaciones entre las múltiples variables que afectan los procesos. Para poder aplicar estas herramientas es necesario plantear objetivos de búsqueda y aplicar una metodología de descubrimiento de conocimiento en bases de datos (Knowledge Discovery in Databases, KDD).

Sin embargo, debido a la alta complejidad del sistema ACS, los objetivos de búsqueda que guían el análisis son múltiples, lo que requiere de una comprensión muy acabada de todo el proceso para poder modelar algunas partes a través de la minería de datos y buscar entonces las relaciones que permiten comprender por ejemplo, porqué y bajo qué circunstancias el sistema falla. El estudio y comprensión del sistema ACS permitirá la identificación de un problema específico para el cual es posible plantear una aproximación basada en minería de datos.

1.2. Objetivo General

- Identificar una problemática y generar una propuesta para aplicar una metodología de minería de datos aplicado a los registro de logs de ACS, que de solución al problema identificado.

1.3. Objetivos Específicos

- Identificar un problema relevante para ALMA y que pueda ser abordado analizando los logs del sistema ACS.
- Establecer los requisitos, objetivos y alcances de la problemática a analizar.
- Plantear una metodología, basada en el proceso KDD, para abordar la problemática identificada.
- Formular un proyecto para resolver el problema planteado.

2. Antecedentes Generales

2.1. ALMA Common Software

El ALMA Common Software(ACS) es la infraestructura de software, con una arquitectura distribuida e integrada, en donde se manejan todos los procesos del observatorio, desde la captura de los requerimientos para la observación hasta la entrega de los datos capturados [2].

Una de las características de ACS es su carácter distribuido, posibilitando que la funcionalidad del sistema pueda implementarse en diferentes lugares geográficos.

Esto hace que ACS sea un sistema heterogéneo, donde diferentes máquinas (hardware) pueden ejecutar software en distintos sistemas operativos y lenguajes de programación. El objetivo de un sistema de estas características es que la comunicación entre clientes y servidores sea transparente, independiente de la arquitectura. Esta filosofía de desarrollo se basa en la separación entre la funcionalidad y la arquitectura técnica.

El propósito del *framework* ACS es:

- Proveer un modelo de programación, para asegurar que una misma función pueda ser desarrollada y ejecutada de la misma forma en cualquier plataforma de desarrollo.
- Provee interfaces para ser implementadas.

ACS provee una filosofía de desarrollo orientado a objetos y los servicios básicos para el cómputo distribuido.

Dentro de estos servicios se encuentran:

- Invocación remota de objetos transparente.
- Manejo distribuido de errores y alarmas
- Manejo distribuido de reporte de eventos (*logging*)

El *framework* ACS está basado en CORBA[2], y está construido en base a código libre.

2.2. Logs

2.2.1. Definición de Log

Un log es un registro o bitácora que indica la actividad de un sistema. Estos registros o eventos son posteriormente almacenados y tienden a responder el qué, el dónde, el cuándo de un evento particular del sistema [3].

Listing 1: Ejemplo un evento XML de log

```
<Debug TimeStamp="2002-10-7T13:44:16.530"
Host="te1.hq.eso.org" Process="baciTestServer" Thread="main"
Context="" File="baciTestClassImpl.cpp" Line="205"
Routine="BaciTestClass::~BaciTestClass">
    Great debug message!
</Debug>
```

Los archivos de Logs son fuentes de valiosa información acerca del estado de un sistema, o de un conjunto de sistemas interconectados. En el listing 1 se puede ver un ejemplo de un registro.

Los Logs a procesar son documentos en formato XML. Estos archivos contienen registros que indican información acerca del estado del sistema en forma de eventos. Los eventos, a su vez, contienen datos específicos que describen los detalles asociados al evento en cuestión. Por ejemplo, el instante de tiempo donde ocurre el evento, la máquina que origina, el proceso asociado, entre otros, y además de una descripción en lenguaje natural generada por un programador.

Todos estos eventos están originalmente clasificados por un tipo que indica el grado de severidad del incidente. Por ejemplo, *Error* , *Warning*, *Info*, *Debug*, *Critical*, etc. [3].

2.3. Pre-procesamiento de datos

2.3.1. Documentos legibles por máquinas

Uno de los mayores problemas en los archivos de logs es que no están diseñados para el procesamiento automático, sino para ser interpretados por operadores humanos. Por lo tanto, es difícil generar un mecanismo automático para clasificar estos eventos en forma sistemática.

2.3.2. Tratamientos de texto para los eventos

Uno de los problemas para poder realizar clasificaciones es que casi todos los eventos son diferentes y es necesario agruparlos según criterios para reducir el número de estos.

Varios de los sistemas de predicción necesitan identificadores numéricos para caracterizar cada tipo de error[4]. En algunos sistemas los eventos incorporan un identificador, y otros simplemente incorporan únicamente información detallada para cada evento, por lo que es necesario realizar algunas de las siguientes operaciones:

- Separar atributos de cada evento.
- Remover números en los mensajes.
- Eliminar atributos vacíos.
- Transformar el tiempo a formato UNIX
- *Stemming* de palabras, (reducción a a la raíz de las palabras)
- Eliminación de conectores en mensajes.

Posteriormente de haber tratado los eventos, un algoritmo de clasificación de texto puede generar *clusters*. Algunos de los algoritmos usados para realizar clasificación de documentos de texto son:

- **Distancia de Levenshtein**: Este algoritmo calcula la distancia de dos textos, según el mínimo de operaciones requeridas para poder igualar ambos textos. Este valor es útil para poder comparar el parecido de dos mensajes de evento.

- **Kmeans - IDF**: Este algoritmo combina dos conceptos diferentes el primero es la matriz inversa de frecuencia de palabras, en otras palabras, la frecuencia de cada termino en un documento, este documento puede ser un mensaje de un evento. El segundo algoritmo kmeans, puede relacionar y agrupar estas frecuencias para determinar clusters con frecuencias similares de los clusters, este algoritmo requiere un tratamiento extra, eliminando conectores y aplicando stemming.
- **Locality-sensitive hashing algorithms**: Este tipo de algoritmos permite reducir la dimensionalidad de los datos. Al generar aproximaciones es posible realizar agrupaciones y comparar la similitud entre dos mensajes. Esta clase de algoritmos son utilizados para crear huellas digitales de documentos de video y audio, entre otras aplicaciones.

2.4. Procesamiento de datos

Luego de haber pre-procesado los registros, es posible realizar el procesamiento de minería de datos para la búsqueda de patrones. Uno de los algoritmos posibles es:

2.4.1. SVM

Las Máquinas de Vectores de Soporte (SVM) abordan problemas de clasificación lineal, mediante la búsqueda de la superficie óptima de separación de dos clases. La superficie óptima, maximiza la distancia entre las dos clases, utilizando los datos más cercanos (vectores de soporte) para construir dicha superficie, como se describe en la Figura 1 [5]. Cuando el problema es no lineal se realiza una proyección de los datos de entrada hacia un espacio llamado "espacio de características", generalmente de alta dimensión. Siempre existe un espacio, por más alta que sea su dimensión, en el que es posible encontrar un hiperplano lineal que separa dos clases, como lo muestra la Figura 1.

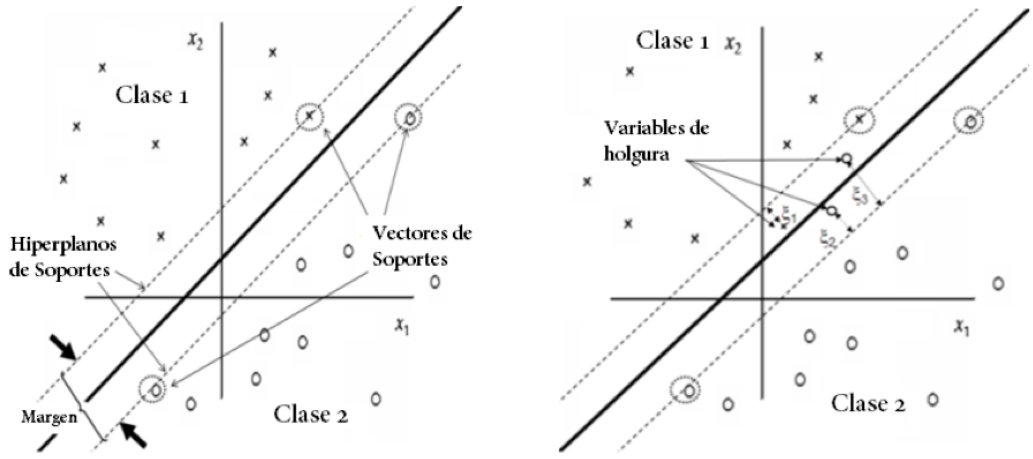

Figura 1: Hiperplano lineal de separación de dos clases: (**a**) Margen máximo (**b**) Margen blando. Los datos destacados con círculos, son vectores de soporte

Los parámetros del hiperplano lineal (w, b) se obtienen mediante un algoritmo de optimización que encuentra la mayor distancia (margen) respecto de los vectores de soporte[6]. Se definen los hiperplanos de apoyo que se muestran en la Figura 1 (a). Por su parte, la Figura 1 (b) muestra la situación de margen blando, o sea cuando se permite que algunos puntos crucen los hiperplanos de soporte. Las variables de holgura, son términos que indican en qué medida un punto se encuentra en el lado equivocado de su respectivo hiperplano de soporte. Para resolver problemas no lineales, se utiliza el "truco del kernel"[6]. La función de transformación, llamada kernel, proyecta el espacio de entrada en el espacio de características, como se puede observar en la Figura 2.

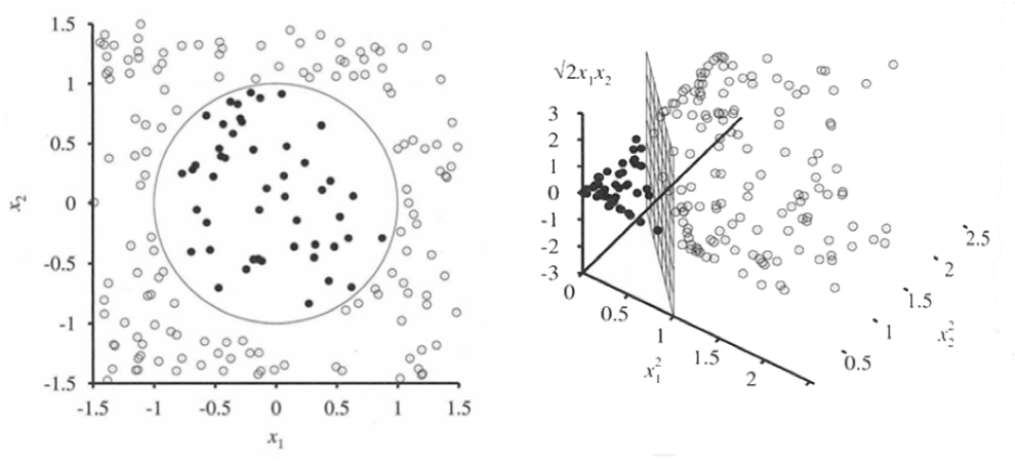

Figura 2: Efecto de la asignación del espacio de entrada en un espacio de características dimensional superior, donde es posible un plano de separación lineal.

3. Materiales y Métodos

3.1. Metodología

El propósito de este trabajo es proponer una metodología de minería de datos para logs de ACS. Para esto utilizaremos el proceso de descubrimiento de conocimiento en bases de datos[7], por sus siglas en inglés **KDD** (Knowledge Discovery in Databases). Este proceso consta de varias etapas:

1. Identificación de una problemática a estudiar.
2. Desarrollo de conocimiento previo del dominio de la aplicación y los objetivos del usuario final.
3. Creación de un set de datos, en dónde se llevarán a cabo las operaciones.
4. Limpieza y pre-procesamiento de los datos, eliminación de ruido, estrategias para el manejo de datos faltantes y análisis de las secuencias de tiempo.
5. Proyección de los datos, búsqueda de características útiles que sean representativas del objetivo del problema. Reducción de los datos, utilizando reducción de dimensionalidad o métodos de transformación para disminuir el número de variables o encontrar una representación invariante para los datos.
6. Implementación del método de procesamiento de datos.
7. Búsqueda de patrones utilizando el método escogido, utilizando una representación de interés para reglas de clasificación, regresión, *clustering*.
8. Interpretación de los datos minados, en caso de que los resultados no sean los esperados es posible volver a iterar cualquiera de los pasos anteriores.

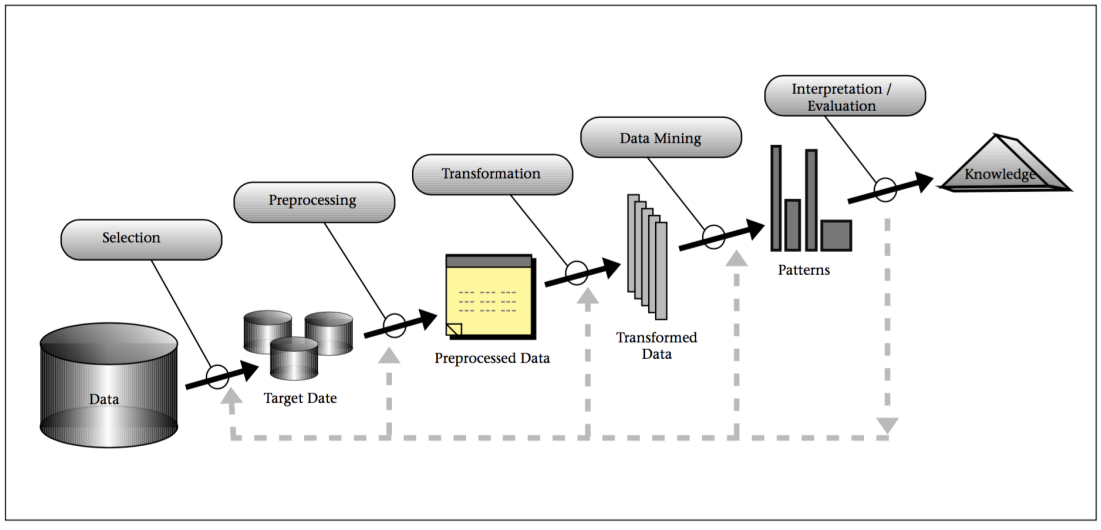

Figura 3: Proceso KDD
Fuente: Fayyad [7]

4. Programa de Trabajo

4.1. Descripción de las actividades

1. Identificar en conjunto con el experto de ALMA una problemática contingente al análisis de logs.
2. Definir los requisitos del problema identificado.
3. En conjunto con el experto de ALMA, establecer características relevantes en los patrones de LOGs.
4. Implementar una herramienta para la representación visual de las secuencias de log.
5. Abordar la problemática utilizando la literatura relacionada con el tema de análisis de logs.
6. Identificar una técnica adecuada para enfrentar la problemática.
7. Desarrollar un proceso reducido de **KDD**, sin iteraciones, basándose en la técnica previamente seleccionada.
   a) Selección de un set de datos y operaciones de pre-procesamiento sobre los mismos.
   b) Extracción de características de las secuencias.
   c) Procesamiento de los datos.
   d) Validación de los resultados.
8. Elaboración del proyecto.

4.2. Carta Gantt

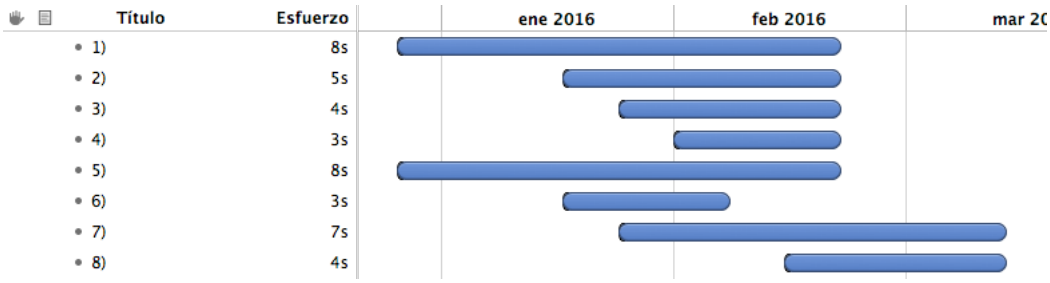

Figura 4: Carta Gantt

5. Nomenclatura

Nomenclaturas utilizadas:

Evento o registro	Se refiere a cada línea del archivo donde se encuentra el LOG
LOG	Documento donde se encuentran todos los eventos
TAG	Identificador numérico (no único) para cada evento
Mensaje	Descripción de un evento, escrito en lengueaje naural
Atributos	Valores que contiene un evento (Tiempo, máquina, etc)
Tipo de evento	Hace referencia a la categoría de incidencia: Trace (2) Delouse Debug (3) Info (4) Notice (5) Warning (6) Error (8) Critical (9) Alert (10) Emergency (11)

6. Bibliografía

Referencias

[1] European Southern Observatory. About alma, 2015. URL https://web.archive.org/web/20151202195912/https://www.eso.org/sci/facilities/alma/about-alma.html.

[2] Gianluca Chiozzi, Birger Gustafsson, Bogdan Jeram, Mark Plesko, Matej Sekoranja, Gasper Tkacik, and K. Zagar. CORBA-based Common Software for the ALMA project. In *Astronomical Telescopes and Instrumentation*, pages 43–54, 2002. doi: 10.1117/12.461036.

[3] European Southern Observatory. Logging and archiving, 2007. URL http://www.eso.org/~almamgr/AlmaAcs/OnlineDocs/Logging_and_Archiving.pdf.

[4] Felix Salfner and Steffen Tschirpke. Error Log Processing for Accurate Failure Prediction. *Wasl*, pages 1–8, 2008.

[5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, 1995. ISSN 08856125. doi: 10.1007/BF00994018.

[6] B Schölkopf, J C Platt, J Shawe-Taylor, a J Smola, and R C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001. ISSN 0899-7667. doi: 10.1162/089976601750264965.

[7] Usama Fayyad, G Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, pages 37–54, 1996. ISSN 0738-4602. doi: 10.1145/240455.240463. URL http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/1230.