MASTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIONES

TRABAJO FIN DE MASTER

Plataforma genérica para el análisis y predicción de logs de Windows a nivel empresarial

Autor: Pablo Sánchez Naharro

Director: Ramiro García Salazar

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

"Plataforma genérica para el análisis y predicción de logs de Windows a nivel empresarial"

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico **2018/19** es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Pablo Sánchez Naharro Fecha: 11 / 06 / 19

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Ramiro García Salazar Fecha: 11 / 06 / 19

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor **D. Pablo Sánchez Naharro**_____

DECLARA ser el titular de los derechos de propiedad intelectual de la obra:

Plataforma genérica para el análisis y predicción de logs de Windows a nivel empresarial,

que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 11 de Junio de 2019

ACEPTA

Fdo. Pablo Sánchez Naharro

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

MASTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIONES

TRABAJO FIN DE MASTER

Plataforma genérica para el análisis y predicción de logs de Windows a nivel empresarial

Autor: Pablo Sánchez Naharro

Director: Ramiro García Salazar

Madrid

PLATAFORMA GENÉRICA PARA EL ANÁLISIS Y PREDICCIÓN DE LOGS DE WINDOWS A NIVEL EMPRESARIAL

Autor: Sánchez Naharro, Pablo.
Director: García Salazar, Ramiro.
Entidad Colaboradora: Devo (http://www.devo.com)

RESUMEN DEL PROYECTO

Se ha desarrollado un sistema genérico para la predicción de eventos en streaming, así como un entrenamiento casi real-time del mismo (1 época de desfase, aproximadamente 2 minutos). Este modelo se ha conectado con la plataforma Devo para la ingesta de eventos y almacenamiento de los resultados. Adicionalmente se ha creado un sistema de visualización del entorno (calidad de los logs) y de las predicciones realizadas

Palabras clave: Devo, TensorFlow, Attention, Streaming, Deep-Learing

1. Introducción

Actualmente existen dos claras tendencias en el mundo de los datos: una de ellas es el procesamiento de eventos en real-time streaming, y la otra el uso de Machine Learning (ML). La combinación de ambas puede crear sinergias en determinados ámbitos de conocimiento. Una de las áreas donde esta combinación resulta especialmente beneficiosa es, por ejemplo, la ciberseguridad.

2. Definición del proyecto

Devo permite el tratamiento de grandes volúmenes de información (100K Events/s). Actualmente se aplica ML partiendo de cada evento individual. Este proyecto pretende extraer información de la relación temporal existente entre diferentes eventos. En concreto, se centrará en el sistema de logs de equipos con sistema operativo Windows.

Se ha desarrollado un sistema para inferir relaciones entre eventos mediante el uso de ML, junto con un dashboard que permita analizar el flujo de información entrante en la plataforma. La información reflejada en el dashboard permite tanto visualizar irregularidades como descubrir situaciones donde el modelo puede fallar por el hecho de ser diferentes.

3. Modelos

Se han aplicado modelos de ML para la identificación de los eventos y para la predicción de estos. Los modelos de identificación están basados en modelos de Autoencoder y el modelo aplicado para la predicción toma como base el modelo Transformer [1].

El modelo se ha implementado desde cero mediante Tensorflow únicamente tomando como base el diseño del modelo [1][2]. Sobre este diseño se han realizado modificaciones para adaptar el problema original para el que se diseñó (traducción de idiomas) al problema de predicción de eventos.

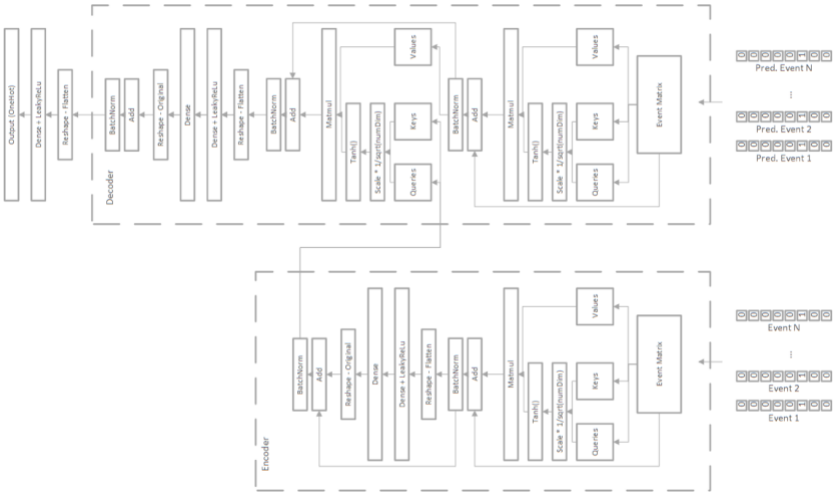

Ilustración 1 – Esquema del modelo Transformer adaptado

4. Dashboard

El dashboard analiza el flujo de eventos recibidos para garantizar que la recepción de eventos se encuentra dentro de lo normal. Para ello obtiene estadísticas de periodos de tiempo anteriores y se comparan con lo ocurrido en la actualidad.

Entre los grupos de análisis se encuentran: eventos por región geográfica, eventos por canal y eventos por tipo.

5. Streaming

El procesamiento en streaming de los eventos presenta un conjunto de procesos ejecutados en paralelo. Por una parte, existe un sistema de predicción que contiene el modelo en su última época completa y genera predicciones en real-time. Por otra, existe un modelo entrenándose continuamente, que se transfiere al predictor al finalizar cada época.

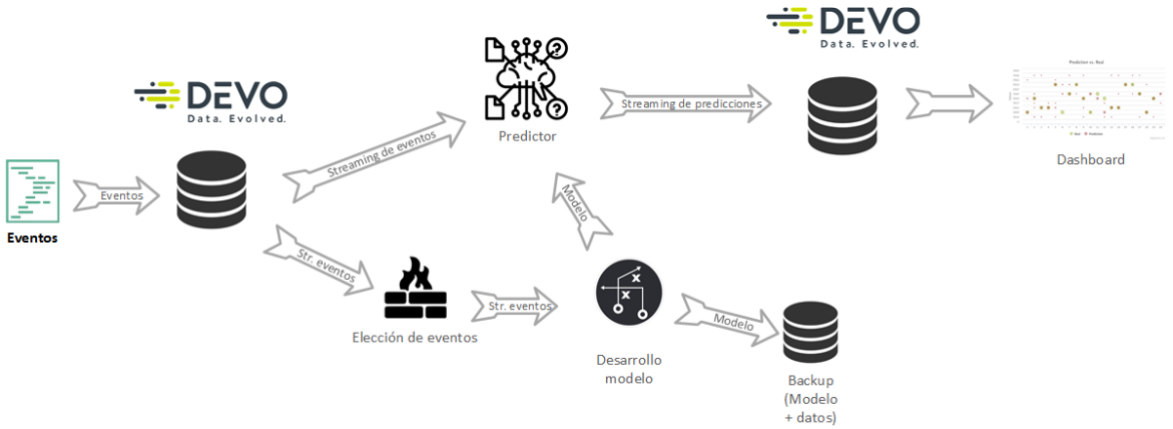

Ilustración 2 - Flujo de eventos en el procesamiento en streaming

Ante la problemática de aparición de eventos no conocidos, se ha creado un sistema capaz de hacer crecer el modelo cuando sea necesario. El nuevo modelo creado estará inicializado con los pesos [3] del último modelo entrenado para no perder el trabajo hecho, y continuará entrenándose con los eventos que llegan.

6. Resultados

Dado que el framework está orientado al procesamiento de eventos en streaming, no es posible realizar una validación del mismo mediante la división del dataset. La cifra obtenida variará en función de la naturaleza de los eventos recibidos anteriormente (con los que el modelo ha sido entrenado) y los que se reciben posteriormente.

Para comprobar el proceso de aprendizaje se ha diseñado la siguiente visualización:

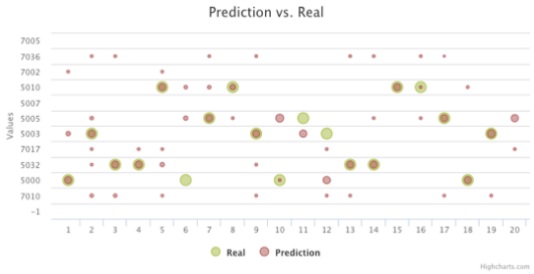

Ilustración 3 – Visualización de la Predicción vs. Eventos ocurridos

7. Conclusiones

Se ha conseguido desarrollar una prueba de concepto para la predicción en streaming de eventos junto con sus probabilidades de ocurrencia. Además, este modelo es capaz de crecer si fuera necesario y de entrenarse con los últimos eventos recibidos.

Este modelo es aplicable a cualquier secuencia de eventos que se inyecte en la plataforma.

8. Referencias

[1] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gómez, A.N.; Kaiser, L.; Polosukhin, I.. "ATTENTION IS ALL YOU NEED". Google Brain, Google Research & University of Toronto. Diciembre, 2017:

https://arxiv.org/abs/1706.03762

[2] Alammar J. "THE ILLUSTRATED TRANSFORMER" Junio, 2018:

http://jalammar.github.io/illustrated-transformer/

[3] Morgan. "TENSORFLOW: SAVING/RESTORING AND MIXING MULTIPLE MODELS" Noviembre 2016:

https://blog.metaflow.fr/tensorflow-saving-restoring-and-mixing-multiple-models-c4c94d5d7125

GENERIC PLATFORM FOR ANALYSIS AND PREDICTION OF WINDOWS LOGS IN COMPANIES

Author: Sánchez Naharro, Pablo.
Supervisor: García Salazar, Ramiro.
Collaborating Entity: Devo (http://www.devo.com)

ABSTRACT

It has been developed a system with the purpose of predicting events in streaming. The system will also be trained in nearly real time (1 epoch delay, approximately 2 minutes). The model has been connected to Devo's system for the input events in streaming and for store the result. It has also been developed a visualization system for measuring the quality of the event log and the predictions done.

Keywords: Devo, TensorFlow, Attention, Streaming, Deep-Learing

1. Introduction

Nowadays, there are two clear tendencies in data world: real-time streaming event processing and Machine Learning (ML). The combination both of them can create synergies in some areas of knowledges. One of the areas where this combination is specially profitable is, for example, cybersecurity.

2. Project definition

Devo is able to deal with large volumes of information (100K Events/s). Nowadays, ML can be applied to each individual event. The aim of the project is to get information from the temporal relationship of the events. In particular, it will be focused on log system from Windows PC.

It has been developed a model for inferring events relationship using ML. In addition to this, a dashboard for analyzing the flow of the events has been programmed. The information displayed on the dashboard allows to both discover abnormalities and situations where the model could fail due to a change in surroundings.

3. Models

It has been applied ML models for the identification of the events and the prediction of them. The identification models applied are based on Autoencoders and the prediction model, on Transformer [1].

The model has been programmed from scratch using TensorFlow. It has only been used the designed model [1][2] as starting point. This design has been modified to adapt the model from the original problem (language translation) to prediction problem.

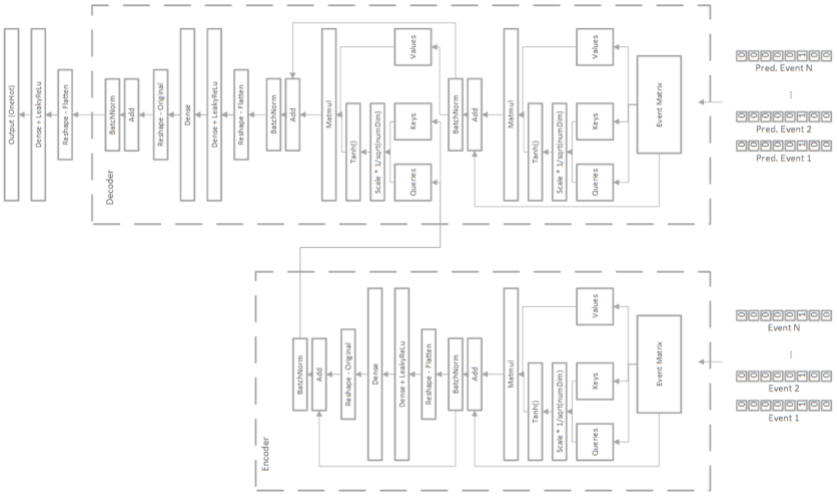

Figure 1 – Adapted Transformed Model schema

4. Dashboard

The dashboard analyzes the inbound event flow to check that the input streaming events are not corrupt. For doing this, statistics from actual periods of time are calculated and tested against the previous flows.

Some of the analysis are: division per geographic region, events per channel and events per type.

5. Streaming

Streaming event processing is based on a bunch of parallel threads. On one hand, there is a prediction system which generates real-time predictions and whose model is copy of the one saved in the ending of the last training epoch. On the other hand, there is a model training continuously; which is transferred to the predictor at the end of each epoch.

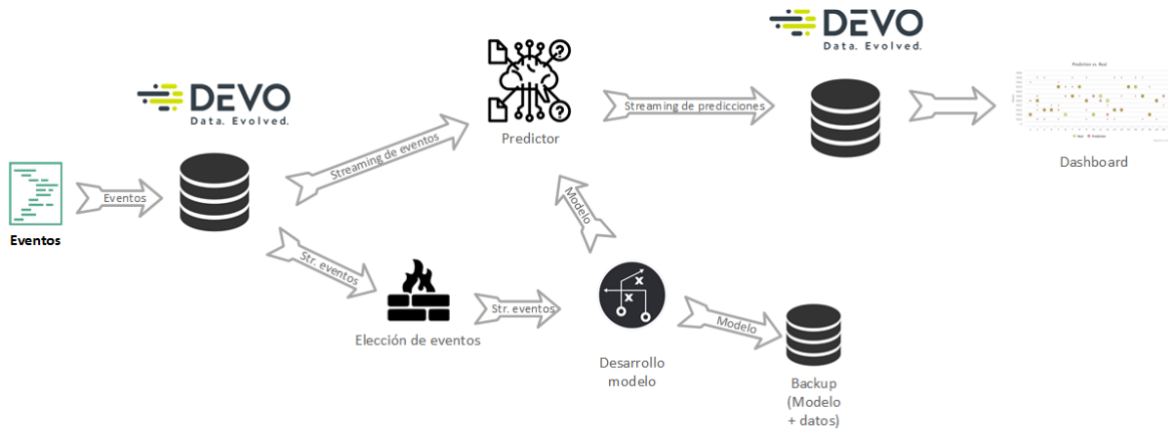

Figure 2 – Flow of events for the streaming processing,

In order to manage the unknown inbounding events, a system for making the model grow has been developed. The new model will be initialized with the previous model weights [3] to avoid the loss of previous model information. Then, it will be trained with the new events.

6. Results

The framework is focused on the processing of streaming events. Due to this, it is not possible to validate the model using the technique of dataset division. The accuracy will change depending on the nature of the previous inbound events (the ones used for training the model) and the ones received later (for predict).

In order to check that the model is learning, it has been developed the following graph:

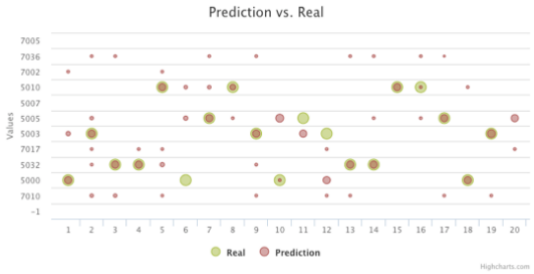

Figure 3 - Prediction vs. Real events graph

7. Outcomes

The project achieves the purpose of developing a proof of concept for a streaming prediction of events and its probabilities to happen. In addition, the model is able to both, growing whenever it is necessary, and be trained continuously with the latest events.

The model can be applied to any sequence of events that is injected in the platform.

8. Referencias

[1] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gómez, A.N.; Kaiser, L.; Polosukhin, I.. "ATTENTION IS ALL YOU NEED". Google Brain, Google Research & University of Toronto. Diciembre, 2017:

https://arxiv.org/abs/1706.03762

[2] Alammar J. "THE ILLUSTRATED TRANSFORMER" Junio, 2018:

http://jalammar.github.io/illustrated-transformer/

[3] Morgan. "TENSORFLOW: SAVING/RESTORING AND MIXING MULTIPLE MODELS" Noviembre 2016:

https://blog.metaflow.fr/tensorflow-saving-restoring-and-mixing-multiple-models-c4c94d5d7125

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

El mundo globalizado supone una apertura de las fronteras, incluido a nivel mercantil. Con esta apertura se crea una situación de competencia voraz entre los diferentes comerciantes. La situación es de tal magnitud, que se está convirtiendo en ficción la capacidad de obtener una ventaja competitiva mediante los procedimientos ordinarios. La competencia se produce cada vez más en regiones de mayor tamaño, y los clientes demandan mayor personalización. Esto supone un reto para los modelos de negocio, donde no sirve con hacer las cosas en una región con competencia limitada, sino que la competencia está asegurada; y por tanto necesitan expandir sus fronteras. En definitiva, nos encontramos en una época en la que el valor añadido de las empresas tiene que basarse en decisiones extraídas de información sobre el comportamiento de los clientes.

Si la ventaja debe extraerse de la información, los datos se convierten en una pieza clave en el modelo de negocio de las empresas. Esta información debe ser almacenada para ser procesada, es decir, tiene que ser susceptible de ser utilizada, lo que supone implementar un Gobierno del Dato.

Toda empresa quiere proteger su negocio; y siendo los datos una de sus claves, realizará inversiones cada vez más importantes en su seguridad. No obstante, se tiene que garantizar el acceso a la información por parte de los empleados, que son quienes generan parte muy valiosa de esa información. Esto supone permitir el acceso desde los ordenadores de uso diario que incorporen las medidas de seguridad pertinentes.

Las medidas de seguridad implementadas mejoran año tras año, pero tienen que permitir al usuario acceder a la información para su explotación. Por tanto, se crea un punto de seguridad dudosa en el blindaje de la información: el usuario.

En la actualidad, el uso de ordenadores se ha convertido en una tarea cotidiana, y en muchas ocasiones estos ordenadores se encuentran manejados por personal no formado en la

seguridad de los equipos. Además, el acceso se realiza generalmente desde ordenadores de oficina sometidos a un uso cotidiano. Esto genera una situación de potencial peligro para la información de las empresas, y por tanto para su modelo de negocio.

Tanto es así, que la mayoría de los ataques hacen uso en algún punto de la denominada Ingeniería Social, para penetrar en los sistemas. Estas técnicas, consisten en hacer que el usuario realice una serie de acciones que se aprovecharán para acceder al sistema.

En este contexto, cobra especial relevancia la existencia de los registros en los ordenadores. Estos registros fueron creados originalmente para hacer una trazabilidad de las averías y fallos que presenta un ordenador y, de esta manera, facilitar la identificación de fallos. Sin embargo, una buena configuración hace que en los registros se almacene cada operación realizada en determinados ámbitos, como pueda ser un fallo o un intento de acceso a información sensible.

Esta información es utilizada para la extracción de las causas de los desastres una vez se ha producido un ataque o una caída de servicio. Esto ocurre en los denominados análisis forenses.

Anteriormente hemos mencionado la importancia que tiene la información en los modelos de negocio. Esto ha impulsado el desarrollo de grandes infraestructuras capaces de manejar cantidades ingentes de información, lo que ha generado otro factor clave.

La combinación de estos dos factores, creciente importancia de la seguridad y desarrollo de infraestructuras de datos masivos; abre la posibilidad de hacer un análisis de los logs de manera continuada con el fin de prevenir ciertas situaciones críticas. Y este es el concepto que se explotará en los siguientes capítulos del documento.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este capítulo proporcionaremos una visión de las soluciones que se dan en la actualidad, así como de las tecnologías utilizadas.

2.1 INTRODUCCIÓN A DEVOINC

Devo es una empresa propietaria de una herramienta para el almacenamiento de datos en el rango de TB, así como su consulta y explotación. Por este motivo, podemos enmarcarla en el ámbito de las tecnologías Big Data.

Para el almacenamiento se dispone de una arquitectura del tipo NoSQL que recibe los eventos en formato Syslog y los almacena en ficheros de texto organizados en una jerarquía de árbol según la categoría, o tabla, a la que pertenece dicho evento.

En el momento de la consulta se accederá a dicho árbol y se recuperarán los ficheros de texto. La información contenida en los mismos se procesará en el momento mediante una serie de parseadores y se convertirá en una tabla.

Esta estructura permite gestionar de manera muy ágil los eventos gracias a que la escritura se realiza de manera inmediata y sin cuellos de botella; y es una arquitectura fácilmente escalable. Con esto se consigue que el tiempo de respuesta dependa únicamente de la cantidad de eventos a procesar en la consulta. Los resultados obtenidos en cuanto a procesamiento de eventos son del orden de 100.000 eventos/segundo.

2.1.1 TIPOS DE CONSULTAS

Devo proporciona tres filosofías a la hora de hacer consultas a la información. Cada una de estas filosofías tiene su centro en una granularidad del dato y una experiencia de usuario diferente.

* **Tabla**: la información se muestra en formato tabla, lo que permite una granularidad muy fina sobre la información. Sobre esta tabla se pueden realizar operaciones y dibujar algunos gráficos simples. La información mostrada se extra directamente de la consulta, a la que se accede con el lenguaje LinQ. Este formato tiene limitaciones a la hora de mostrar información de múltiples tablas, así como para personalizar la visualización para realizar un análisis de granularidad gruesa.
* **Dashboard**: Permiten agrupar diferentes visualizaciones de manera que se pueda extraer información genérica de los datos almacenados de una manera muy ágil. Esta opción permite una personalización rápida de los gráficos. Sin embargo, presenta limitaciones en las visualizaciones ya que solo pueden introducirse gráficos incluidos en la plataforma.
* **Aplicaciones Verticales**: Consiste en un sistema de plug-in para la aplicación. Estos complementos realizan un vitaminado la aplicación mediante un framework de JavaScript (JS) y HTML + CSS atacando directamente a los servicios API REST de la base de datos. Gracias a esta característica permite una personalización casi infinita.

Para este proyecto explotaremos las capacidades de las tablas para realizar el análisis exploratorio de los datos, y las capacidades ofrecidas por las aplicaciones verticales para no estar limitados en cuanto al desarrollo.

2.2 PREDICCIÓN

Para la realización de predicciones basadas en Machine Learning existen dos principales tendencias: LSTM [2][5][18][20] y Attention [1][4].

2.2.1 LSTM

LSTM (Long-Short Term Memory) hace referencia un tipo de neuronas con memoria. Esto es especialmente beneficioso ya que se hace pasar una secuencia de eventos, y un lazo de

realimentación permite tener en cuenta eventos ocurridos con anterioridad al evento introducido en un momento dado.

Este tipo de neurona fue desarrollada para mantener el estado en un momento dado. Podemos ver su estructura en la *Figura 1* . Se aprecian dos bloques principales: un primer bloque que comienza con C_{t-1}, que recibe el estado anterior y realiza una serie de modificaciones sobre este para construir C_t, correspondiente con el estado siguiente.; y un segundo bloque que se encarga de definir tanto las operaciones a realizar (f_t e i_t), como la salida que dará la neurona en dicho instante h_t (no es la salida que propagará el estado). Estas operaciones consisten en calcular cuanto se mantendrá el estado (f_t) y cuanto se actualizará con los nuevos pesos (i_t).

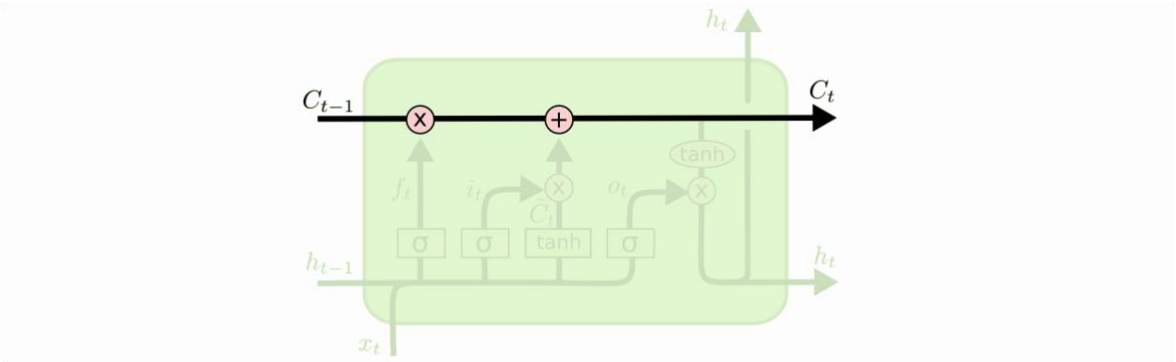

Figura 1 - Estructura interna de una neurona LSTM

Si aplicamos este tipo de neurona a una red neuronal, se mantendrá el estado entre una muestra y otra. Debido a la capacidad de mantener dicho estado, a esta red se la denomina Recurrent Neuronal Network (RNN). En la *Figura 2* podemos ver como es el esquema temporal de una red neuronal. En a parte izquierda se aprecia el lazo de realimentación, encargado de mantener el estado. Además, en el eje vertical podemos ver el flujo de entrada y salida de información. Si nos fijamos en la parte derecha, podemos ver el flujo de información representado en el tiempo. Todas las representaciones hacen referencia a una misma neurona, mostrando como fluye la información entre los diferentes estados, es decir, entre los diferentes instantes de entrada de información.

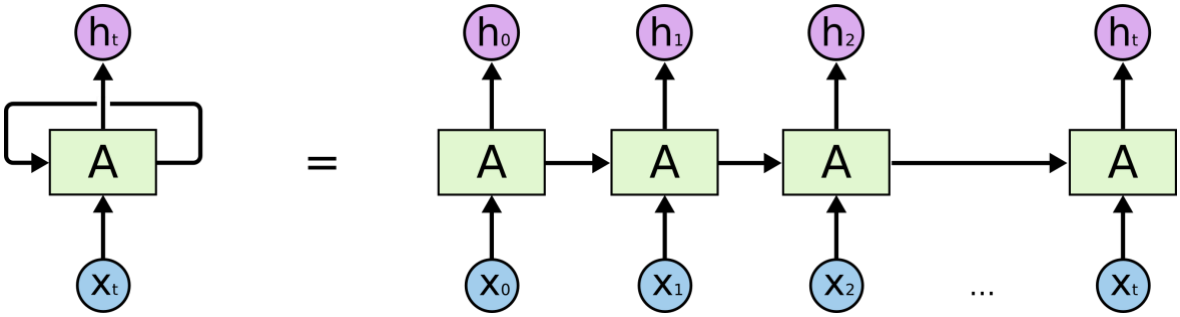

Figura 2 - Esquema de configuración de una neurona LSTM en una RNN

Este concepto puede llevarse también en el sentido anti-temporal [9] si se dispone de todos los elementos de la secuencia al mismo tiempo. Es decir, también introduciremos información del futuro. Este tipo de redes se conocen como Bidirectional Recurrent Neuronal Network (BRNN). En la *Figura 3*, podemos ver como están formadas por dos capas de LSTM: una que transfiere la información en sentido temporal y otra que la transmite atrás en el tiempo. Este tipo de redes tienen gran aplicación en reconocimiento de texto [15] ya que tienen en cuenta tanto la información como posterior. Sin embargo, en el caso que aplica a este proyecto no es de utilidad ya que no se dispone de información a priori.

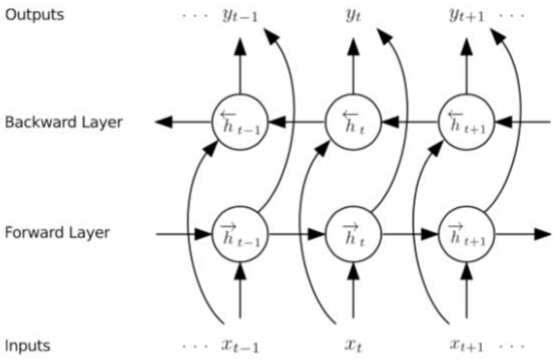

Figura 3 - Esquema de una BRNN

La funcionalidad de RNN cubre justo nuestro caso de uso, sin embargo, es bastante lenta en el aprendizaje ya que la secuenciación de eventos dificulta el entrenamiento paralelo. Para que un evento realmente sea entrenado conforme a su contexto tienen que hacerse pasar los

eventos en la secuencia determinada para que el lazo de realimentación se corresponda con un contexto similar al que ocurriría en la realidad [13].

Además, este modelo supone tener un modelo para cada entidad (en nuestro caso ordenadores, aunque se explicará con mayor detenimiento en el desarrollo del proyecto). Ya que se necesita mantener el estado entre un evento y el siguiente.

Este tipo de redes han sido utilizadas con éxito en otras aplicaciones de secuenciación de eventos como la predicción de enfermedades [8].

2.2.2 ATTENTION

El concepto de *Attention* consiste en identificar cuales son las partes clave de la secuencia de eventos precedente. De esta manera, se puede dar mayor relevancia a estos eventos frente a aquellos que aporten menos información.

En la *Figura 4* se puede ver cual es la estructura de una red propuesta por desarrolladores de Google. Esta estructura está compuesta por una serie de Encoders y Decoders seguidos de una capa de lineal que hace la elección de la predicción en función de los valores de atención (o importancia) calculados por la red.

Originariamente esta red se diseñó para la traducción de textos de un idioma a otro. Esto hace que se tengan que mantener dos secuencias de histórico paralelas. Una de ellas será la secuencia en el idioma original y otra la secuencia en el idioma ya traducido, de manera que se realice una doble función. Por un lado, se realizará la traducción en cuanto a texto (basándose en el histórico original), y por otro se elegirá la palabra, dentro del las que tengan significado similar, que mejor encaje con la traducción proporcionada.

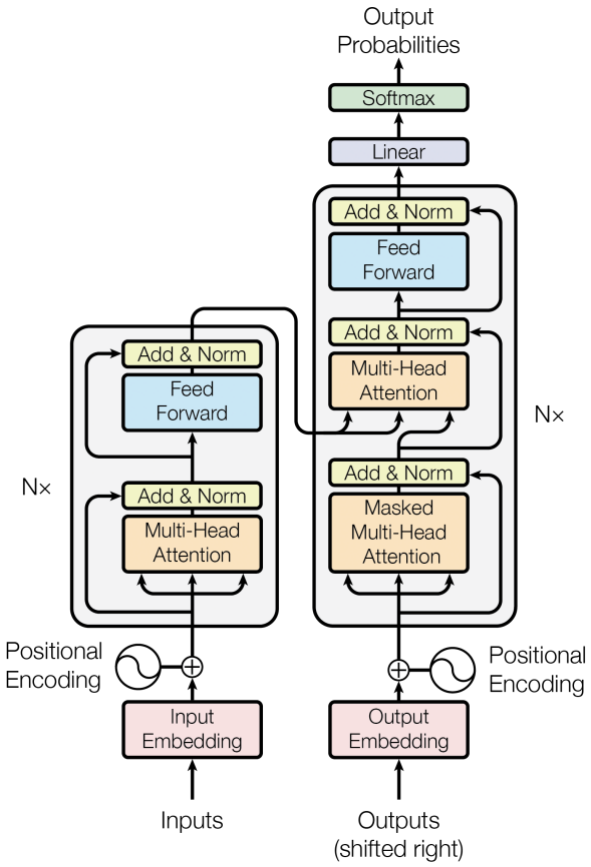

Figura 4 - Esquema de la red neuronal "Transformer"

Si nos adentramos en un Encoder o Decoder, encontramos el esquema de la *Figura 5*, que nos define como está estructurado un bloque de Attention. En esta parte de la red se encuentran unos parámetros calculados que se denominan Keys (K), Queries(Q) y Values (V), lo que recuerda a una base de datos. El flujo de información consiste en aplicar una serie de máscaras y operaciones sobre estos valores para predecir cual será la siguiente salida.

En el esquema original, se colocan múltiples bloques en paralelo creando un Multi-Head Attention. Esta configuración puede verse en la *Figura 6.*

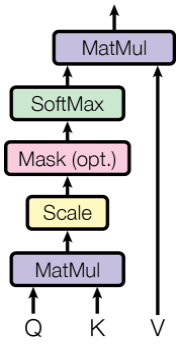

Figura 5 - Esquema de un bloque Attention

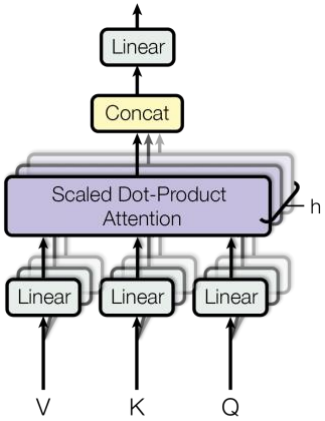

Figura 6 - Esquema de un bloque Multi-Head Attention

Más adelante, en el capítulo *6.2.2*, realizaremos un análisis más detallado de esta red para poder adaptarla a nuestro caso de uso.

Al igual que la LSTM, cubre nuestro caso de uso; pero soluciona el problema de mantener el estado. En este tipo de redes la entrada consiste en un batch de eventos anteriores, sobre los que se calcula la predicción de salida. De esta manera se puede integrar un mismo modelo para las diferentes entidades.

Como beneficios añadidos, se puede desarrollar un modelo de mayor envergadura (no habrá que replicarlo para cada entidad); se reduce el uso de memoria y se permite compartir el conocimiento de manera que dos secuencias similares en diferentes entidades generarán una misma predicción. Este ultimo punto se contrapone al comportamiento de las LSTM, cuya necesidad de mantener el estado supone que cada una aprenderá de su entidad.

Pese a que las redes de Attention suponen claras mejoras, el estado del arte es mucho más extendido en las LSTM. Por tanto, decidimos evaluar ambos modelos para estudiar el de desempeño.

2.3 LENGUAJES DE PROGRAMACIÓN

2.3.1 HTML Y CSS

Dado que se va a desarrollar una extensión para la interfaz web de consultas, se convierte en necesario el uso de HTML y CSS para generar el contenido de la interfaz.

HTML (HyperText Markup Language) es un lenguaje diseñado para describir y estructurar el contenido de las páginas web. Cubre la necesidad de tener un lenguaje descriptivo que permita a los servidores proporcionar información a los navegadores y que estos la interpreten. Dentro de este lenguaje existen los denominados estilos.

La popularización y el aumento del uso de los estilos hacen que el código se desestructure rápidamente. En este punto surgen los CSS (Cascading Style Sheets), que permiten extraer la configuración de los estilos a documento aparte de manera que ambos se mantengan estructurados.

2.3.2 JAVASCRIPT

JavaScript es un lenguaje de programación no tipado utilizado por excelencia en los navegadores Web. Este lenguaje permite dotar de dinamismo a las páginas HTML.

Además, la parte de consultas compatible con la aplicación hace uso de procesado en la parte del cliente. Esto permite proporcionar una experiencia de usuario fluida gracias a que se prescinde de la conexión de la red a excepción de la obtención de información.

Ambos motivos justifican el uso de JavaScript para el desarrollo del interfaz del proyecto.

2.3.3 PYTHON

Python es un lenguaje que dispone de gran cantidad de librerías que facilitan el desarrollo de aplicaciones. Unido a que se trata de un lenguaje no tipado permite alcanzar elevadas velocidades de desarrollo.

Además, se pretende hacer uso de TensorFlow, cuyo principal lenguaje de desarrollo es Python.

2.3.4 TENSORFLOW

Se trata del ecosistema de Google para el procesado de operaciones matemáticas con tensores y con ello, el entrenamiento de redes neuronales.

Al igual que Python se configura como centro a la hora de desarrollar librerías de investigación, Tensorflow proporciona un backend para herramientas que facilitan la labor del desarrollo de redes neuronales; como es el caso de Keras.

Otro de los puntos clave de TensorFlow es la integración con diferentes tipos de procesamiento:

* *CPU*: Procesamiento paralelo aprovechando todos los núcleos del procesador.
* *CPU con características especiales (SSE4.1, SSE4.2, AVX, AVX2, FMA, etc)*. Estas características permiten realizar operaciones de manera más eficiente, o incluyen operaciones adicionales a las estandarizadas de la CPU; lo que permite una ejecución con un mayor rendimiento.
* *GPU:* Dado que las redes neuronales habitúan a usar una serie de operaciones simples, pero en gran cantidad y con capacidad para paralelizar, se obtiene gran beneficio con la carga de este proceso sobre la GPU. Estas unidades integran numerosas ALU (Unidades Aritmético-Lógicas) con operaciones limitadas, pero con gran capacidad de procesamiento paralelo; lo que las convierte en un elemento fundamental para optimizar el rendimiento de las redes neuronales.
* *TPU:* Unidades diseñadas expresamente para el procesamiento de operaciones de Tensorflow

2.3.5 KERAS

La convergencia de las redes neuronales se produce cuando se dan una serie de propiedades matemáticas. En caso de vulnerar ciertas propiedades, surgirán problemas a la hora de

realizar el modelo. Esto hace de la programación en TensorFlow un procedimiento tedioso y delicado.

Este contexto genera la necesidad de crear un interfaz a más alto nivel que permita la popularización de las redes neuronales. Este nicho es el que Keras se encarga de cubrir.

En el proyecto se usará Keras para la realización de ciertas redes neuronales. Sin embargo, se intentará maximizar el código en TensorFlow para tener un mayor control sobre las operaciones realizadas a nivel de personalización y debug.

Capítulo 3. ESTADO DE LA CUESTIÓN

Actualmente la seguridad se consigue infiriendo reglas mediante la correlación de datos en un análisis forense. Posteriormente, estas reglas se transfieren a los IDS y a los SIEM. Estos sistemas son muy complejos y constituyen una rama entera de desarrollo.

El fin de este proyecto no consiste en desarrollar un sistema IDS/SIEM completo, sino que pretende crear una herramienta complementaria a estos sistemas. Debe ser capaz de realizar una predicción genérica para la plataforma Devo. En el contexto de la seguridad, el predictor inferirá automáticamente patrones de comportamiento por lo que supondrá un complemento en la extracción de patrones mediante análisis forense.

Para el desarrollo y prueba, se aplicarán los logs de tarjetas de red para predecir eventos como puedan ser reinicios de esta ya que se encuentran altamente correlados y no existirá una limitación en el aprendizaje por la naturaleza de la información.

Una cualidad clave debe ser la fácil extensibilidad a diferentes áreas, es decir, debe proporcionar un Framework de análisis en streaming para cualquier información que se inyecte a la plataforma únicamente modificando el entrenamiento de la red neuronal de predicción.

3.1 SOLUCIONES ACTUALES

Actualmente el core del negocio de Devo se sitúa en el almacenamiento de datos junto con una serie de herramientas de visualización y consultas de la información. La plataforma Devo ofrece la opción de desarrollo de aplicaciones (parecido a un plug-in) para la visualización de datos con un mayor nivel de personalización.

El proyecto se basará en esta opción para desarrollar una aplicación que permita realizar una predicción basándose en los eventos recibidos, que se verá acompañado de un análisis del contexto para asegurar que se dan las condiciones necesarias para la predicción.

Podemos encontrar diferentes áreas de exploración en función de los dos bloques principales del proyecto: visualización de la información y predicción.

3.1.1 VISUALIZACIÓN DE INFORMACIÓN EN DEVOINC

La plataforma Devo dispone de tres modalidades a la hora de presentar la información:

* Tabla: la información se muestra en formato tabla, lo que permite una granularidad muy fina sobre la información. Sobre esta tabla se pueden realizar operaciones y dibujar algunos gráficos simples. Este formato tiene limitaciones a la hora de mostrar información de múltiples tablas, así como para personalizar la visualización para realizar un análisis de granularidad gruesa.
* Dashboard: Permiten agrupar diferentes visualizaciones de manera que se pueda extraer información genérica de los datos almacenados de una manera muy ágil. Esta opción permite una personalización rápida de los gráficos. Sin embargo, presenta limitaciones en las visualizaciones ya que solo pueden introducirse gráficos incluidos en la plataforma.
* Aplicaciones Verticales: Consiste en un sistema de plug-in para la aplicación. Estos complementos realizan un vitaminado la aplicación mediante un framework de JavaScript (JS) y HTML + CSS atacando directamente a los servicios API Rest de la base de datos. Gracias a esta característica permite una personalización casi infinita.

3.1.2 PREDICCIÓN

A la hora de realizar predicciones existen dos principales vertientes:

* Modelos clásicos de Machine Learning (ML): consiste en una combinación de diferentes modelos de ML para extraer información de los datos sobre los que se

quiere hacer la predicción. En este bloque podemos encontrar algoritmos y técnicas como Gradient Boosting, Random Forest, etc.

* Basado en redes neuronales: se basan la consecución de diferentes modelos de redes neuronales mediante un diseño de alto nivel. Se diferencian principalmente de los anteriores en que el conocimiento será adquirido, generalmente, gracias a un modelo de aprendizaje supervisado acompañado del algoritmo de back-propagation.

Los modelos clásicos tienen la ventaja de tener un mayor nivel de interpretabilidad, mientras que los basados en redes neuronales suelen tener un mayor nivel de precisión a cambio de sacrificar la interpretabilidad.

Capítulo 4. DEFINICIÓN DEL TRABAJO

El análisis de logs en Devo se encuentra actualmente en una fase en la que se extraen métricas de los diferentes datos para que un perfil especializado pueda monitorizar la existencia de diferentes anomalías y con ello prever eventos y tomar medidas. Esta monitorización dota de una gran ventaja a las empresas reduciendo el tiempo de reacción ante diferentes amenazas.

Existe una parte del proceso altamente automatizada. Esta fase representa el cálculo de métricas sobre los millones de logs para dar una visualización sobre el histórico de eventos y facilitar la detección de anomalías.

Sin embargo, la elección de los procesos realizados sobre los datos para extracción de métricas, así como la elección de métricas se realiza de forma manual basándose en experiencias previas. Las métricas intentan cubrir parámetros que indiquen que se produce una anomalía, pero todo basado en el conocimiento de los expertos.

En resumen, se pretende extraer información sobre los eventos ocurridos para predecir posibles eventos cuando se produzcan condiciones similares. En el caso de este proyecto se hará recaer la elección de métricas y monitorización sobre una red neuronal. Se espera obtener relaciones entre los eventos no obtenidas mediante los métodos clásicos.

4.1.1 OBJETIVOS DEL PROYECTO

El proyecto consiste en el desarrollo de una aplicación vertical que contenga un dashboard y un predictor de eventos.

Como características clave se encuentra:

* Procesamiento de datos en streaming. El predictor recibirá un flujo de datos continuo y en real-time y predecirá cual es el siguiente evento. En futuras aplicaciones la

distancia temporal de predicción se verá determinada por los datos elegidos para el entrenamiento, de manera que tendrá que fijarse con antelación a la creación del modelo neuronal.

* Procesamiento de datos en batch (para la obtención de medidas generales). Con el fin de obtener métricas robustas que permitan visualizar el contexto del predictor, se utilizarán los eventos contenidos en un rango temporal para la realización de los cálculos.
* Generación de alertas. La aplicación ha de integrarse con el sistema de alertas de Devo de manera que el usuario pueda configurarlas.

El principal añadido de la aplicación vertical es el predictor de eventos. La fiabilidad de esta predicción está supeditada a la calidad de los datos de entrada. Por este motivo se desarrollará un dashboard que permita tener una visión general del flujo de eventos de entrada.

Aunque el proyecto se centra en desarrollar un framework general que permita hacer predicciones para un conjunto genérico de datos, de manera que se pueda replicar la aplicación para diferentes tipos de problemas con un simple cambio de los datos de entrada; se utilizará un conjunto de datos determinado para el entrenamiento del modelo. En este caso se ha elegido un set de datos sobre errores en tarjetas de red. Estos datos presentan una alta correlación, de manera que los resultados no se vean altamente limitados por los datos utilizados.

La descripción anterior hace referencia al proyecto base. Adicionalmente se plantean dos ciclos de mejora que se desarrollarán en función de la complejidad requerida para la realización de predicciones. Esta complejidad no es fácilmente estimable debido al uso de redes neuronales, que tienen una interpretabilidad baja y por tanto cada mejora contiene un elevado nivel de incertidumbre tanto temporal, como de éxito.

Como primer ciclo de mejora incluimos la creación de visualizaciones que permitan correlar fácilmente el histórico de predicciones con los eventos realmente acontecidos. De esta forma

se podrá hacer un estudio más detallado de en que partes falla el predictor, dando pie a nuevos ciclos de mejora.

En segundo lugar, se implementará un aprendizaje en streaming. Este aprendizaje deberá ser supervisado por un operador que decida si el evento es relevante y por tanto debe ser incluido en la información del predictor. Un evento deberá ser considerado relevante según el impacto que haya tenido en el modelo de negocio, así como en función del contexto en el que se ha producido (eventos en un entorno de pruebas, simulaciones, etc). Existe el riesgo de que esta función genere deficiencias en la predicción, pero a cambio genera un sistema de guía que permite al operario priorizar los eventos sobre los que se quiere hacer mayor hincapié.

Capítulo 5. CONTEXTO DEL MODELO

5.1 SISTEMA DE LOGS DE WINDOWS

El sistema de logs de Windows permite hacer un seguimiento de los eventos que se producen en el ordenador tales como el acceso a diferentes niveles de permiso, la ejecución de programas o secuencias de errores producidas.

5.1.1 INTRODUCCIÓN AL SISTEMA DE LOGS

Para acceder a los diferentes logs del sistema debemos ejecutar el "Visor de eventos". Dentro de esta aplicación existe una sección denominada "Registro de Windows". En este punto podemos comenzar a explorar los eventos. La Figura 7 muestra el interfaz para la exploración de eventos.

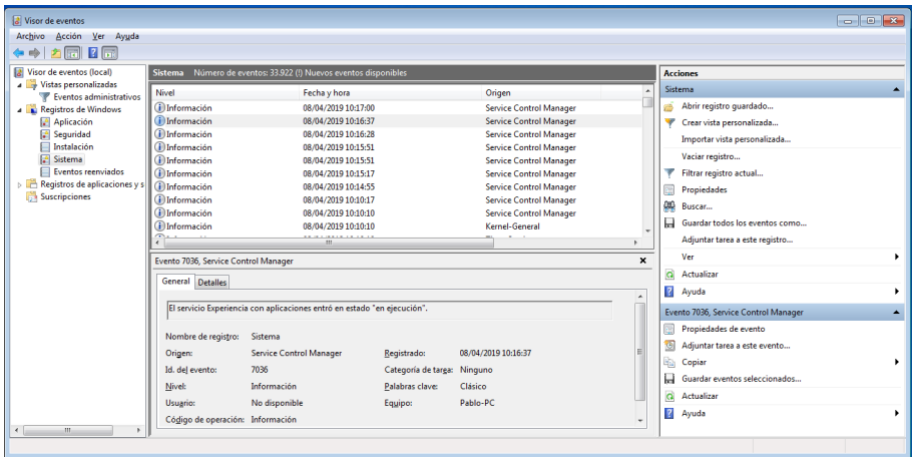

Figura 7 - Visor de Eventos de Windows

Cada evento contiene una serie de campos generales que permiten situar el evento nivel general (motivo, gravedad, etc.) y otros particulares que proporcionan información específica del evento. Los campos clave que nos permitirán hacer un estudio de los eventos son los siguientes:

* *Canal del evento*: Indica a nivel general cual es la naturaleza del evento. Existen cuatro canales por defecto: Aplicación, Seguridad, Instalación y Sistema. Se pueden crear nuevos canales de eventos para adaptar la agrupación de estos al caso de uso.
* *Nivel del evento*: Hace referencia a la severidad de la acción o evento que reporta el registro. Ej.: información, crítico, advertencia, … Este campo se divide a su vez en dos que son redundantes: uno de ellos indica cual es la severidad en forma numérica y otro a nivel descriptivo.
* *Host*: Equipo que genera el evento. Este campo es almacenado en el servidor, pero no se puede ver desde la interfaz ya que este solo refleja el registro de un único ordenador.
* *Origen*: Especifica la aplicación que lanza el evento.
* *ID del evento*: Indica exactamente cual es el evento. Este campo es especialmente importante ya que identificará biunívocamente el tipo de evento. En caso de que el ID se repita para eventos no similares, deberá crearse un campo que lo identifique. En el *Capítulo 6.* se propone un método que permite identificar los eventos. Este método presenta una serie de ventajas respecto a la asignación secuencial de IDs.
* *Descripción del evento*: explicación en lenguaje natural el evento.
* *EventData*: contiene información específica del evento.

Esta información se almacena en formato estructurado XML. A continuación, mostramos un evento de cierre de sesión.

```
Error! Hyperlink reference not valid. <Event
xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
Error! Hyperlink reference not valid. <System>
<Provider Name="Microsoft-Windows-Security-Auditing" Guid="{54849625-5478-4994-
A5BA-3E3B0328C30D}" />
<EventID>4647</EventID>
<Version>0</Version>
<Level>0</Level>
<Task>12545</Task>
<Opcode>0</Opcode>
<Keywords>0x8020000000000000</Keywords>
<TimeCreated SystemTime="2019-04-03T11:46:42.693800000Z" />
<EventRecordID>13379</EventRecordID>
<Correlation />
<Execution ProcessID="540" ThreadID="5860" />
```

```
<Channel>Security</Channel>
<Computer>Pablo-PC</Computer>
<Security />
</System>
<EventData>
<Data Name="TargetUserSid">S-1-5-21-724919297-2408379368-3543393363-1000</Data>
<Data Name="TargetUserName">Pablo</Data>
<Data Name="TargetDomainName">Pablo-PC</Data>
<Data Name="TargetLogonId">0x6ce52</Data>
</EventData>
</Event>
```

5.1.2 ARQUITECTURA DE ENVÍO DE DATOS

Los datos se envían a la plataforma a través de un conmutador denominado Relay. Este conmutador se encuentra en la red cliente y contienen los certificados de autenticación para el envío. La arquitectura propuesta permite dividir en dos tramos la gestión del envío de eventos: por una parte, la empresa cliente controla sus eventos internos, pudiendo reducir el nivel de encriptación si la información no sale de la propia empresa o reutilizando el sistema de encriptación sin tener que compartir cierta información con Devo. La segunda parte de la comunicación se gestionará completamente desde el proveedor una vez se hayan intercambiado los certificados por primera vez.

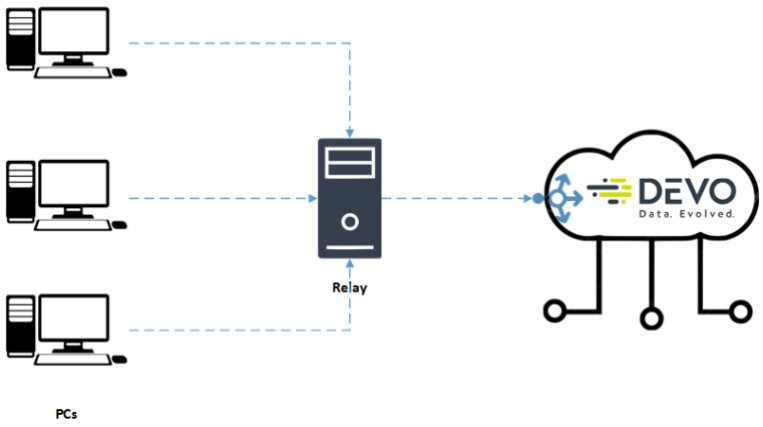

Figura 8 - Arquitectura de envío

5.1.3 NXLOG

NXLog es una tecnología que permite automatizar el envío de eventos de ordenadores a un servidor. Con esta herramienta se realiza la recolección de eventos para su posterior almacenamiento en la plataforma Devo.

5.1.3.1 Configuración de NXLog

La configuración deberá almacenarse en un fichero .conf que definirá el funcionamiento de la herramienta. Esta configuración debe cumplir los siguientes requisitos:

* Conversión a Syslog para garantizar el funcionamiento nativo con Devo.
* Conversión a JSON para permitir la retrocompatibilidad con el sistema ya desplegado.
* Envío a través del relay.

Se realiza la configuración de las rutas de acceso al programa, así como para el acceso a los diferentes módulos de la aplicación.

```
#define ROOT C:\Program Files\nxlog
define ROOT C:\Program Files (x86)\nxlog

Moduledir %ROOT%\modules
CacheDir %ROOT%\data
Pidfile %ROOT%\data\nxlog.pid
SpoolDir %ROOT%\data
LogFile %ROOT%\data\nxlog.log
```

Cargamos los módulos para la conversión a JSON de los eventos, así como el módulo Syslog que permite generar eventos compatibles.

```
<Extension json>
    Module      xm_json
</Extension>

<Extension _syslog>
    Module      xm_syslog
</Extension>
```

Elegimos de que canales vamos a leer información

```
<Input in>
  Module            im_msvistalog
       ReadFromLast True
       Query               <QueryList>\
                                 <Query Id="0">\
                                       <Select Path="Application">*</Select>\
                                       <Select Path="System">*</Select>\
                                       <Select Path="Security">*</Select>\
                                 </Query>\
                           </QueryList>
</Input>
```

Definimos el punto de destino

```
<Output relay>
    Module      om_tcp
    Host        <IP_relay>
    Port        13000
    Exec        $Message = to_json(); $SourceName="box.win-nxlog";
to_syslog_bsd();
</Output>
```

Por último, definimos el flow que seguirán los eventos

```
<Route 1>
    Path        in => ssl_devo_eu, file
</Route>
```

5.2 ALMACENAMIENTO DE DATOS EN DEVOINC

Devo es una plataforma orientada al almacenamiento de información no estructurada. El almacenamiento de datos en Devo se realiza mediante ficheros de texto que se completan a medida que se reciben nuevos eventos.

Estos eventos deben enviarse en formato syslog, de forma que pueda extraerse cierta información clave referente al envío (parte en negrita). El resto de información del evento se incluirá en el campo mensaje.

```
<14>2017-12-11 07:54:08.923 macbook-pro-de-pablo-2.local=10.1.0.196
(naharrosp)my.app.wd.arp.test1: 1469013877|MAC|1.1.1.1|IPV6|Device1
```

Para darle más utilidad a los diferentes campos se aplica una serie de parser sobre los eventos en raw cuando el usuario solicita acceso a los datos. La información resultante se extrae en forma tabular y se envía a los diferentes agentes de consulta sobre los que está actuando el usuario.

5.2.1 ARQUITECTURA DE ALMACENAMIENTO

La arquitectura de la plataforma (*Figura 9*) destaca por la existencia de dos puntos de acceso. Un punto de acceso comunica con el balanceador de carga y se utiliza exclusivamente para la ingesta de datos en formato syslog. El otro punto de acceso se utiliza para acceder a la información de manera transparente al almacenamiento de los archivos.

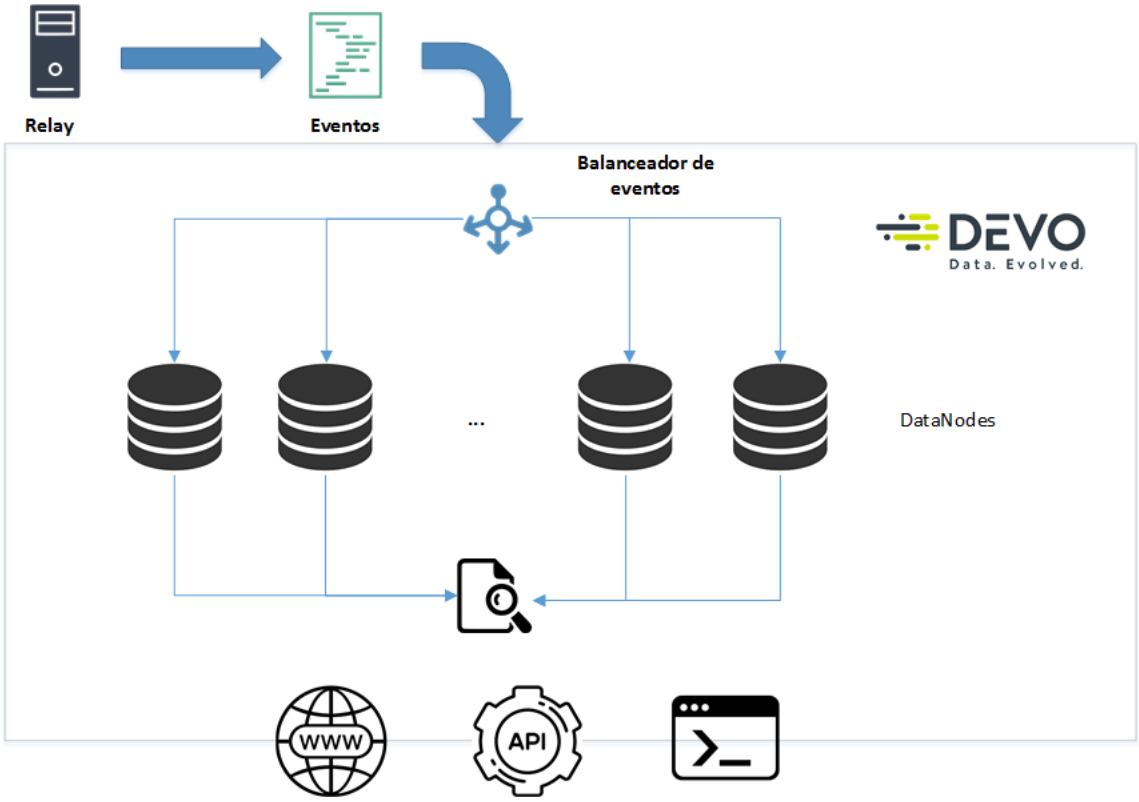

Figura 9 - Arquitectura del almacenamiento y consulta de datos

5.2.1.1 Balanceador de eventos

Este sistema dota a la plataforma de escalabilidad horizontal ya que se encarga de la gestión del almacenamiento de información de forma transparente al resto de elementos externos. Interactúa con la base de datos de localización para mantener consistentes los metadatos de localización de los eventos.

5.2.1.2 Estructura de archivos

Los datos se almacenan en las diferentes máquinas directamente sobre el sistema de archivos. Existen dos principales niveles de organización:

* **Fecha**. A su vez se subdivide en Año-Mes-Día
* **Directorio de almacenamiento**. Contiene dos subniveles de trabajo: propietario y subcarpeta. Dentro de esta subcarpeta se almacenan los diferentes archivos de texto

que contienen los eventos en raw. Suele utilizarse un matching 1 a 1 entre la tabla y la subcarpeta.

Esta organización facilita tareas como el backup y el acceso de eventos en paralelo. Esto se justifica ya que un parámetro clave a la hora de realizar consultas y agrupaciones es el rango temporal sobre el que se quiere hacer.

5.2.1.3 Motores de consulta

Los motores de consulta sirven de punto de acceso a los diferentes servicios para la adquisición de información. Deben acceder a la base de datos de localización para obtener la totalidad de los eventos. Una vez localizados los nodos donde se almacena la información, se realiza el acceso. Por último, se ejecutan los parser para estructurar la información de cara a los sistemas de consulta.

En esta arquitectura, el concepto tabla queda desvirtuado pese a estar presente en los diferentes métodos de acceso. Por simplicidad, podemos asemejar una tabla de los sistemas de consulta, al procesado con un parser específico.

Existen tres principales sistemas de acceso:

* Sistema de consultas Web: aplica las transformaciones necesarias para interactuar con la información desde el interfaz web.
* API Rest: permite la interacción orientada a servicios automatizados tales como aplicaciones verticales.
* CLI (Command Line Interface): permite acceder a los datos internamente, evitando ciertas transformaciones y controles; y facilitando el desarrollo de herramientas.

5.2.1.4 Despliegue en local

Dentro del proyecto, se desea hacer una demostración completa del sistema. Para ello deben cubrirse tanto la fase de ingesta como la fase de analítica y procesado. Por este motivo, realizamos el despliegue de la *Figura 10* en local. No obstante, para las fases de analítica y procesado se interactuará con la plataforma en Internet ya que contiene los datos.

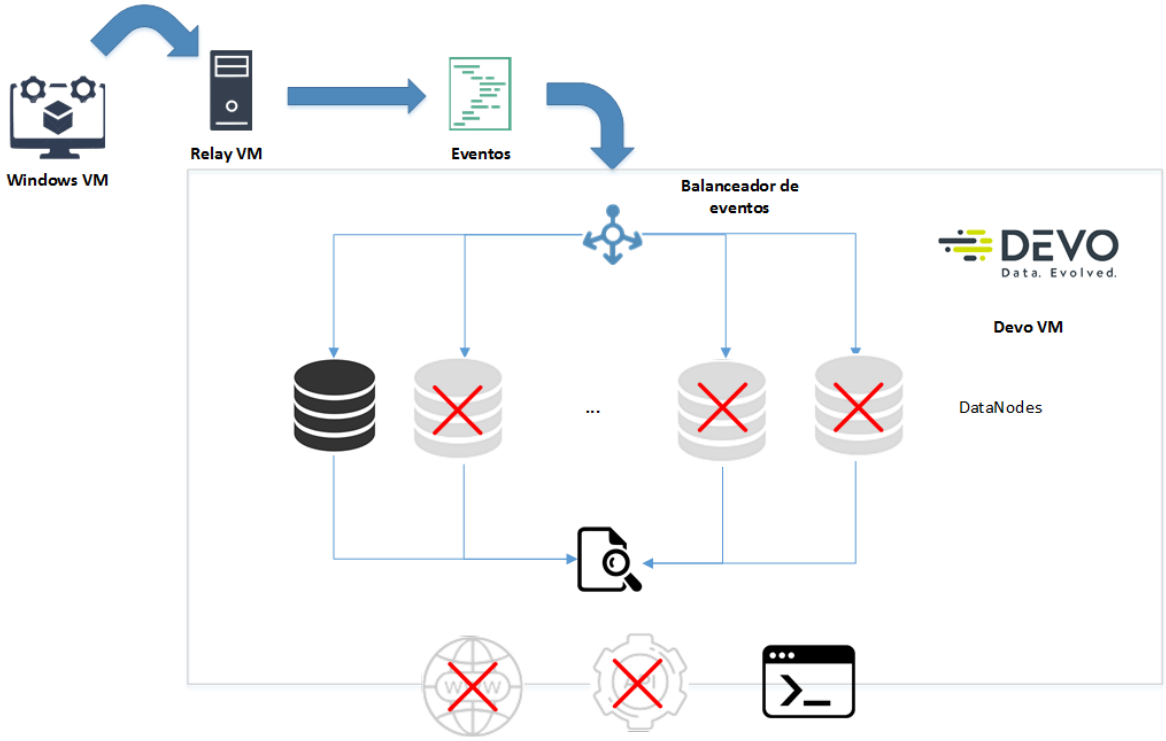

Figura 10 - Arquitectura del despliegue en local

Este esquema reduce el número de DataNodes a 1 por cuestiones de recursos, y limita el acceso a los datos a CLI.

Para la simulación se desplegará una máquina virtual tipo Windows sobre la que se ha instalado NXLog con la configuración explicada en el *Capítulo 5*. Esta máquina envía eventos a una segunda máquina virtual que realiza las funciones del Relay. Por último, se conecta el Relay con una tercera máquina virtual que contiene una simplificación de la plataforma Devo.

Dentro de la máquina de Devo, construimos los parser para que se pueda acceder a la información desde el CLI. En el código presentado a continuación, se muestra un ejemplo de parser. El parser se ha simplificado por motivos didácticos y de confidencialidad.

El siguiente fragmento de texto crea la definición de la tabla para tener una descripción de la misma antes del procesado de datos. Se utiliza para construir las visualizaciones y comprobar las consultas previamente al procedimiento de parsing.

```
table box.win_nxlog {
  signature {
    eventdate {eventdate};
    host = split(hostchain, "=", 0);

    Hostname: str {dstar};
    Channel: str;
    EventID: int4;
  }

  parsing "box/win_nxlog"
    with box.win_nxlog;
}
```

El siguiente fragmento define el propio parser que actuará sobre la información almacenada.

```
fileformat(
  "box.win_nxlog",
  locating('\n'),
  sequential(
    eventdateUTC(),
    terminated(field("hostchain"), " "),
    then(
      seq(
              jsonobj(
                when("Hostname", jstr("Hostname")),
                when("EventID", jnumber("EventID", int4, int4)),
                when("Channel", jstr("Channel")),
              ) // End jsonobj message
            ) // End then message
          ),
          field("rawSource")
        )
      ),
      field('rawTagged')
    )
  )
```

5.3 ANÁLISIS DE DATOS

En la plataforma Devo se almacenan un total de 53 características para los eventos de Windows de manera genérica. Aunque dentro de estos eventos existen particularidades que permiten extraer más o menos características; el predictor debe basarse en las principales para cumplir el requisito de ser generalista.

5.3.1 DEFINICIÓN DE LA IGUALDAD ENTRE EVENTOS

Al existir tal cantidad de características, debe definirse un nivel donde los eventos se consideren iguales. Es decir, si buscamos una igualdad con todos los campos; un evento realmente similar ocurrido en dos equipos diferentes se considerará diferente puesto que el host será diferente. Esto impediría la transferencia de conocimiento a nivel general. Por otra parte, si relajamos excesivamente la condición de clustering podríamos encontrarnos con la situación contraria.

Estudiando los datos, hemos encontrado dos campos principales a la hora de definir un evento: EventID y descripción. Al filtrar individualmente por uno de estos dos campos obtenemos características comunes como la severidad del error y el tipo de evento.

Debe elegirse cual es el nivel de granularidad que se quiere obtener. En función del campo, se han desarrollado dos soluciones:

* **Acceso por EventID**. Esta forma no tiene en cuenta datos de granularidad fina como la ruta de los datos a los que se accede; solo se tendría encuentra el evento de acceso. En caso de hacer referencia a un evento en el que estos datos no se producen (como fallos, etc.); no se produce una reducción en la cantidad de información. Como ventaja, presenta mayor rendimiento.
* **Acceso por descripción**. Con este método si se tienen en cuenta los criterios de granularidad fina. Sin embargo, requiere un pre-procesado del evento para asignarle un identificador. Para esto se ha desarrollado en el Capítulo 6. un sistema capaz de

asignar una codificación OneHot a una descripción. Esta codificación clasificará los nuevos eventos como una combinación de los ya existentes.

5.3.2 DEFINICIÓN DE UNA SECUENCIA DE EVENTOS

Para poder realizar una predicción correcta se debe buscar una forma de hacer clustering sobre las secuencias de eventos. Esto equivale a determinar campos definen una secuencia de eventos.

La elección del primer nivel de agrupación es crítica ya que todos los eventos que lleguen a este filtro serán considerados de la misma secuencia. Esto supone que, si se quiere extraer información de eventos con valor diferente en este nivel, es decir, diferente secuencia; deberá desarrollarse un método de interacción entre secuencias.

Partiendo del EventID, elegimos como primer nivel de agrupación el SourceName. Con esto, generamos secuencias independientes del proceso que las genera y del thread al que pertenece; lo que supone un aumento de la incertidumbre en la predicción (varios procesos atacan a una misma secuencia), pero también se consigue una correlación entre procesos (una situación o secuencia de eventos, dividida en subprocesos del mismo source seguirá perteneciendo a la misma secuencia).

En la *Figura 11*, se ha utilizado una visualización en forma de grafo secuencia para estudiar la cardinalidad. Vemos como se relacionan las características host, Channel, sourceName y eventID. Comenzamos analizando el gráfico de derecha a izquierda.

La primera secuencia de cruces representa la relación entre EventID (derecha) y el SourceName (Izquierda). Al construir la visualización, cada nodo ID se acerca lo más posible a su SourceName correspondiente. Por este motivo parece que los múltiples ID convergen a una misma SourceName. No obstante, se producen ciertas conexiones que parecen romper esa convergencia generalizada. Estos cruces indican que un mismo ID está asociado a múltiples SourceNames, por tanto, SourceName es un elemento necesario a la hora de definir un flujo.

Aplicando esta misma lógica, comprobamos en el bloque del medio como se relacionan los SourceName con los Channel. En este caso los Channel agrupan múltiples SourceNames, pero es bastante habitual que los SourceNames notifiquen a diferentes Channel. Así que también se puede considerar Channel como una clave. En este caso, vamos a descartar este elemento como clave puesto que queremos explorar una interacción entre la existencia de secuencias de eventos de cualquier tipo y eventos críticos (generalmente agrupados en el canal de seguridad).

Por último, las relaciones de la izquierda muestran la existencia de eventos desde un host a un determinado canal. En este caso el Host si se va a considerar como clave.

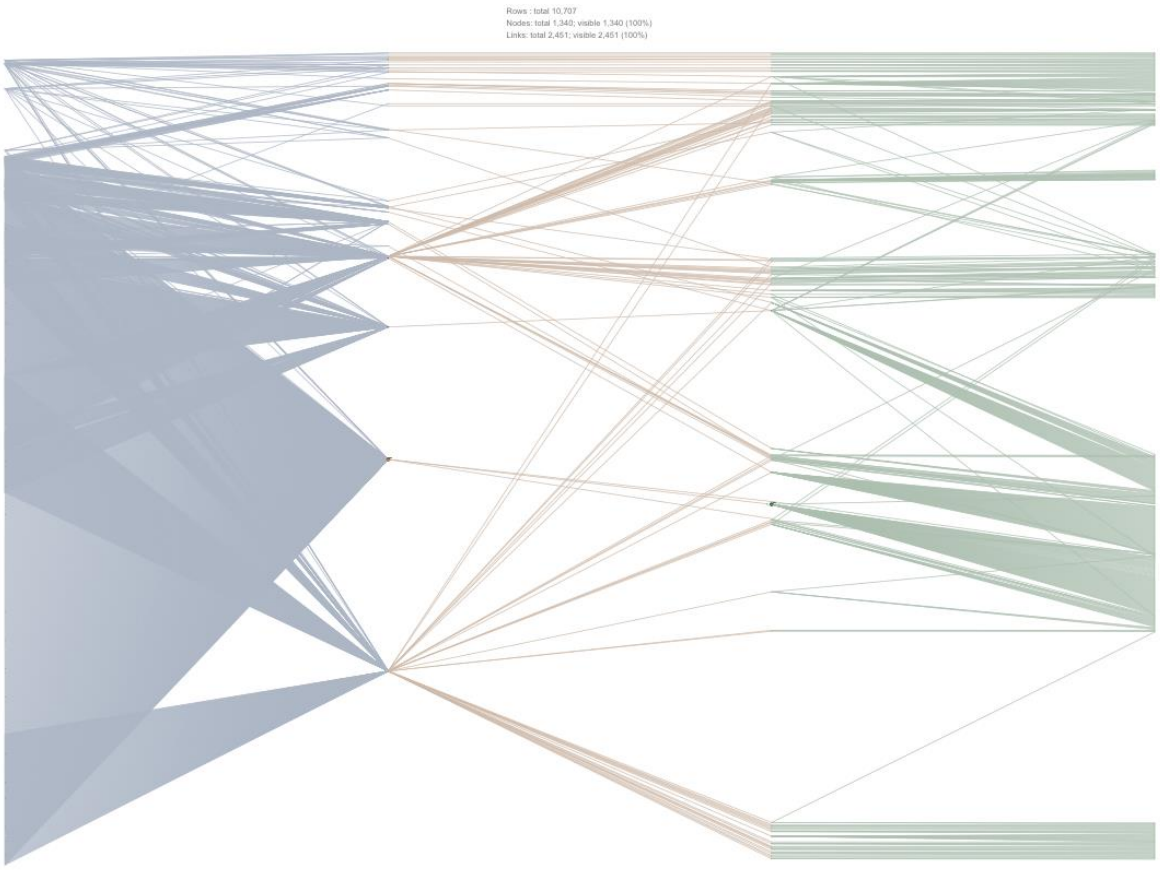

Figura 11 - Grafo de relación (Host -- Channel -- SourceName -- EventID)

Si filtramos para un solo host (*Figura 12*), vemos más claramente la relación de pertenencia entre unos elementos y otros. En este caso desaparecen los cruces entre diferentes Channel-SourceName y diferentes SourceName-ID dado que se ha reducido en gran medida el número de eventos, pero no por ello pueden hacerse simplificaciones de cardinalidad (no se puede decir con carácter general que un grupo contiene a otro y por tanto eliminarlo de la clave).

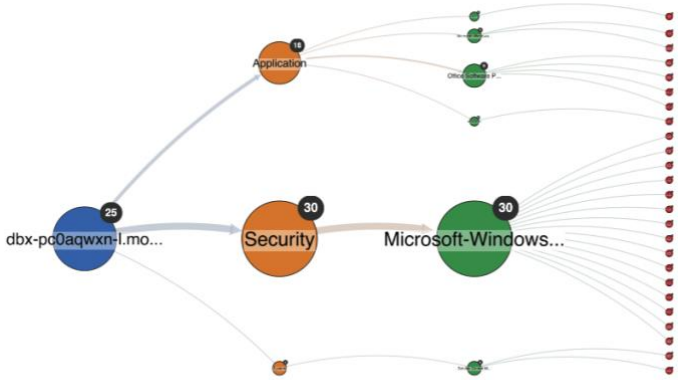

Figura 12 - Grafo de relación (Host -- Channel -- SourceName -- EventID) para un host

En resumen, hemos definido una secuencia de eventos por su Host + SourceName.

5.3.3 ELECCIÓN DEL FLUJO DE EVENTOS

Para el desarrollo del predictor debemos elegir una secuencia con alta correlación de eventos para no limitar su rendimiento por la aleatoriedad de los datos. Tras en análisis de múltiples secuencias, se ha elegido la secuencia de logs correspondientes a errores en la tarjeta de red.

Estos eventos presentan la correlación mostrada en la *Figura 13.*

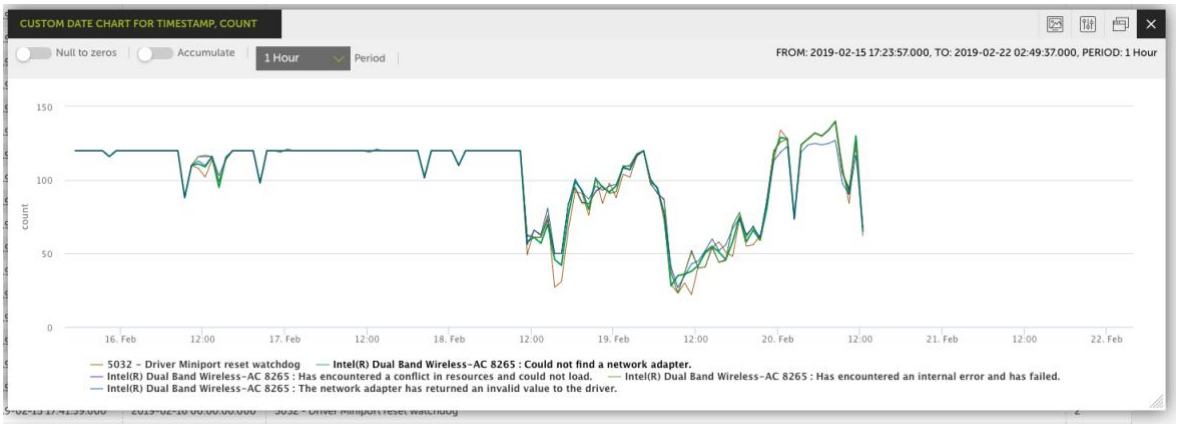

Figura 13 - Representación temporal de la cantidad de eventos de diferente tipo para el SourceName de tarjetas de red

Se observa como la cantidad de logs correspondientes a los eventos presenta una gran proporcionalidad. Esto significa que, si ocurren N eventos tipo A, en ese mismo rango temporal, sucederán aproximadamente N eventos tipo B. Y por tanto parecen tener una gran relación.

Para la obtención del gráfico se ha ejecutado la siguiente query:

```
from box.win_nxlog
  where SourceName = "Netwtw06",
    host = "my-host-A.com",
    Message = "Intel(R) Dual Band Wireless-AC 8265 : Has encountered a conflict
in resources and could not load." or Message = "Intel(R) Dual Band Wireless-AC
8265 : Has encountered an internal error and has failed." or Message = "Intel(R)
Dual Band Wireless-AC 8265 : The network adapter has returned an invalid value to
the driver." or Message = "5032 - Driver Miniport reset watchdog" or Message =
"Intel(R) Dual Band Wireless-AC 8265 : Could not find a network adapter."
  group every 30m by Message, timestamp
  every -
  select count() as count
```

5.4 DESARROLLO DEL DASHBOARD

Una de las características de la aplicación consiste en analizar la normalidad de los logs recibidos. Para ello hay se ha planteado un análisis de la cantidad de logs que se reciben en función de diferentes parámetros. Se considera un rango anormal de eventos aquellos que se sitúen fuera de los percentiles para los últimos tres días contabilizados cada media hora, es decir, tomamos 72*24*2 muestras para el cálculo de los valores.

Con este análisis se pretende comprobar que los logs introducidos en el predictor no han sufrido una variación que afecte a la fiabilidad de este. La *Figura 14* representa una vista preliminar del dashboard desarrollado.

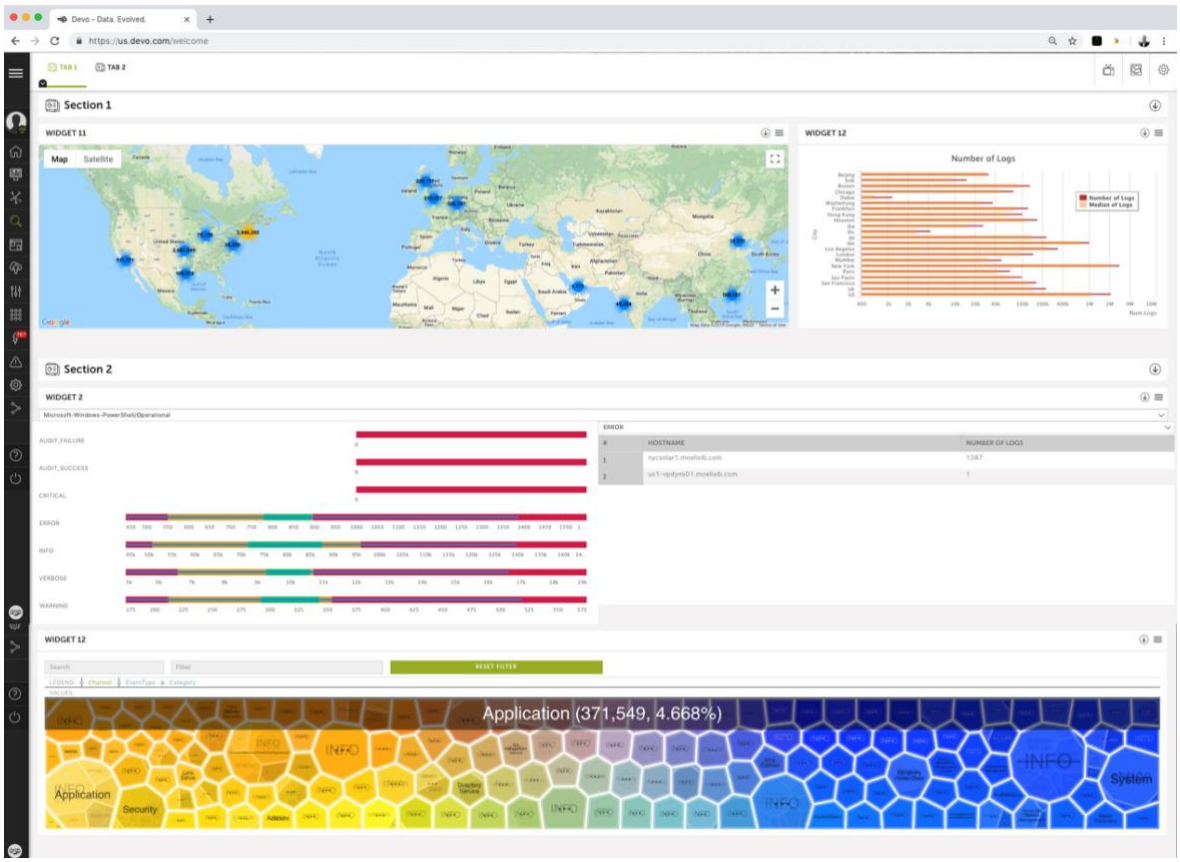

Figura 14 - Dashboard de análisis general

5.4.1 DEFINICIÓN DE LOS GRÁFICOS

5.4.1.1 Mapa de localización de logs

El widget de la *Figura 15* permite visualizar los logs provenientes de cada zona del mundo. Con esto se pretende visualizar de manera efectiva la aparición de zonas desde las que no se deberían recibir. Esta situación puede deberse, bien a la conexión de algún empleado sin VPN o bien a un intento de acceso no autorizado.

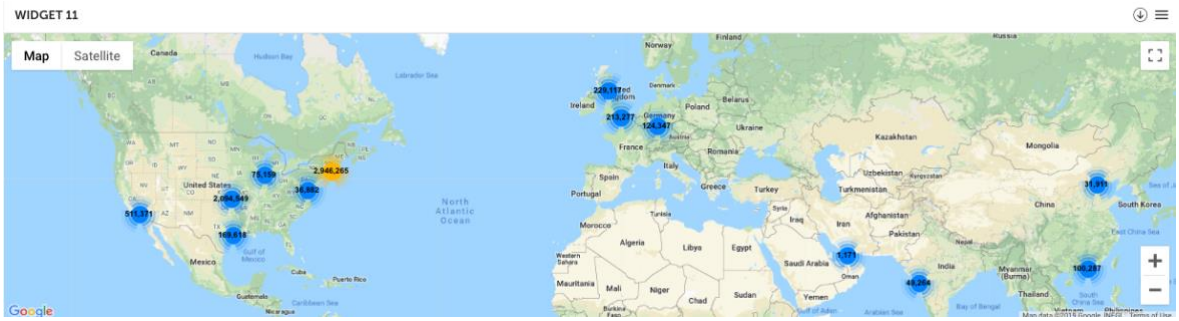

Figura 15 - Gráfico de localización de logs

5.4.1.2 Logs por localización

La *Figura 16* muestra la cantidad de logs respecto a la mediana de logs calculada con las muestras agrupadas según lo explicado anteriormente, 72*24*2. Esta métrica muestra que localizaciones están recibiendo logs por encima o por debajo de lo habitual.

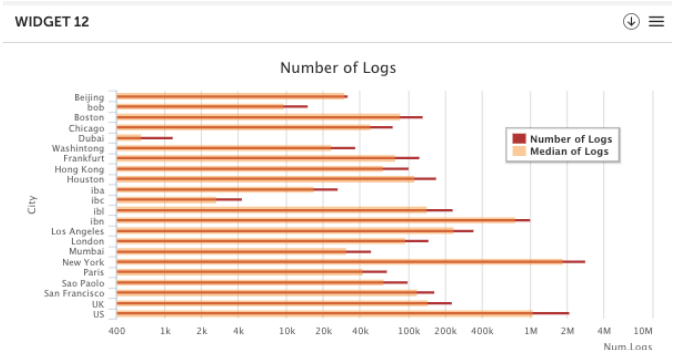

Figura 16 - Número de logs por localización

Como existen localizaciones con diferentes usos, la cantidad de eventos varía múltiples unidades de escala. Por este motivo se ha decidido usar una representación logarítmica.

5.4.1.3 Cantidad de logs por canal y tipo

A continuación, realizamos un análisis de la cantidad de logs por canal y tipo. Podemos ver un análisis de ejemplo en la *Figura 17*. En el gráfico de la izquierda se analiza la cantidad de logs para un tipo de canal elegido, y dentro del canal se subdivide el gráfico en los diferentes tipos de eventos existentes.

En este caso se ha desarrollado un nivel de cantidad tipo semáforo. Existe una franja de color verde en la que la cantidad se considera de normalidad y corresponde con los valores entre los percentiles 0.25 y 0.75. A ambos lados se encuentra una franja amarilla en la que se considera que existe cierta desviación en la cantidad de logs introducidos, pero no se considera un valor crítico. Estos valores corresponden a los valores definidos por los rangos de percentiles: 0.05-0.25 y 0.75-0.95.

En la parte derecha del gráfico se presenta un ranking de los equipos que más logs generan para un canal y tipo de evento determinado. De esta forma, si se detecta una anomalía en los eventos del gráfico de la izquierda, puede analizarse si se trata de un problema generalizado o localizado en un equipo,

Figura 17 - Análisis de eventos por canal y tipo

5.4.1.4 Diagrama Voronoi por canal, tipo de evento y categoría

Por último, se desea visualizar rápidamente cuales son los canales que más eventos generan, así como el tipo de evento y categoría de este. El diagrama sirve de guía a la hora de elegir el canal susceptible de análisis en el gráfico tipo semáforo descrito anteriormente.

Así mismo, puede profundizarse de manera interactiva en el canal y el tipo de evento para disponer de más información a la hora de detectar anomalías.

La *Figura 18* muestra un ejemplo de uso del gráfico:

1. Se detecta un canal que ha variado sustancialmente su tamaño respecto a situaciones anteriores.
2. Accedemos al canal para ver que tipos de evento están produciendo dicho cambio. En este caso se observa un incremento en la cantidad de eventos de acceso.
3. Por último, se detecta la aparición de accesos a objetos especiales.

Figura 18 - Ejemplo de uso del gráfico Voronoi

5.4.2 ARQUITECTURA DEL CÓDIGO

El código combina el uso de ciertos gráficos existentes en la aplicación y de gráficos de librerías externas como Highcharts. Además, debe ser compatible con la integración en la plataforma; por ello tiene que utilizar el framework existente. La *Figura 19* muestra la arquitectura desarrollada.

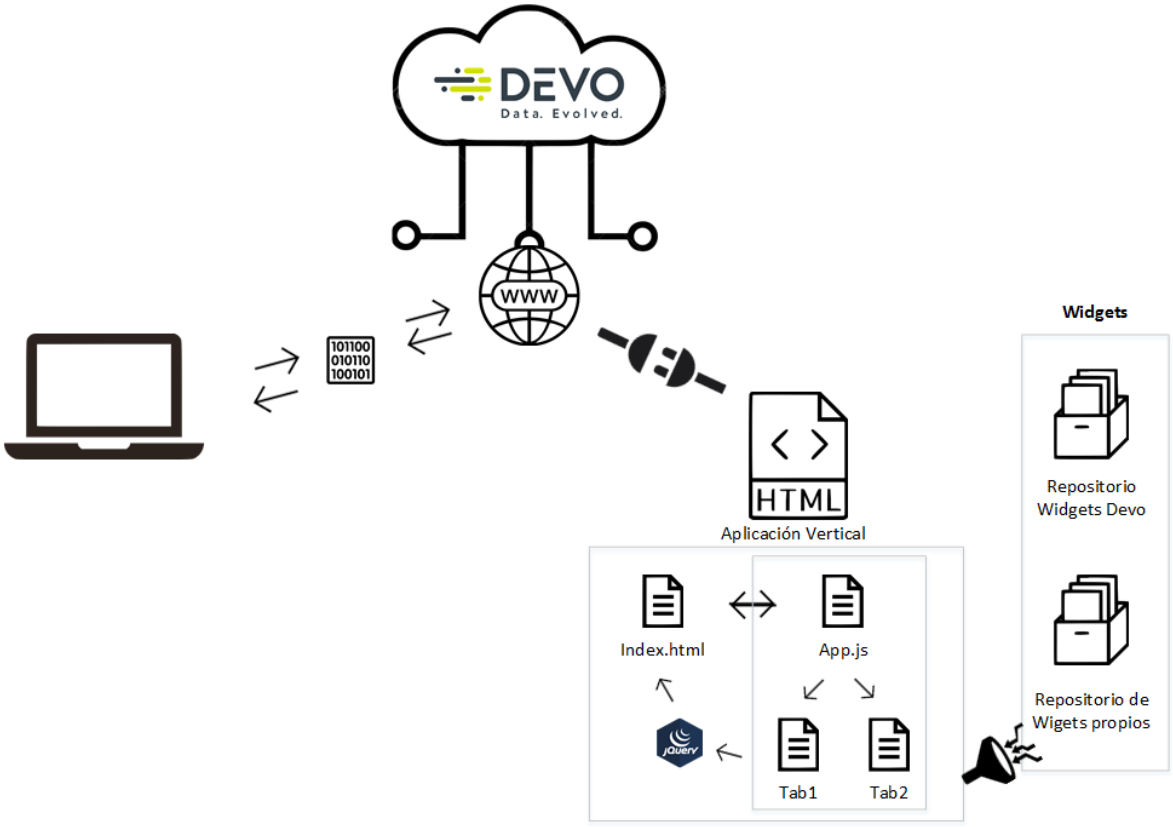

Figura 19 - Arquitectura del código de la aplicación vertical

El dispositivo cliente se conecta directamente con la plataforma Devo. Dentro de esta plataforma, existe una sección en la que se permiten conectar aplicaciones desarrolladas para complementar el análisis de los datos. Este procedimiento es transparente al cliente una vez ha activado la integración de la aplicación seleccionada. Es el servidor el que se encarga de realizar la integración mostrando una pestaña más, correspondiente a la aplicación, dependiendo del cliente.

La aplicación consiste en un HTML dinamizado mediante JS. El código HTML, junto con un diseño CSS, definen la estructura de los componentes de la aplicación. Posteriormente se realiza una sincronización a través de jQuery de los elementos definidos en HTML con los contenidos dinámicos definidos en el JS.

Para la creación de contenido dinámico se usan gráficos definidos previamente, que pueden pertenecer, bien al Framework de Devo; o bien estar definidos de manera independiente. En este caso los gráficos independientes se han creado usando la librería de Highcharts.

Capítulo 6. DESARROLLO DEL MODELO

6.1 MODELO PARA GENERACIÓN DE IDS

En este capítulo se propone un modelo para la creación de IDs de los eventos basándose en su descripción. Entre las dos modalidades existentes para la identificación biunívoca de tipos de eventos se encontraban el EventID y su descripción. El campo EventID es numérico, por lo que puede ser inyectado directamente en el predictor con una serie de transformaciones mínimas. Sin embargo, el campo descripción está compuesto de texto; lo que hace necesario un procesado previo a la asignación de IDs.

En la asignación de IDs, se pretende que los logs con un texto muy similar pertenezcan al mismo ID de evento, pero aquellos con diferencias más acentuadas pertenezcan a ID diferentes.

Estas diferencias pueden extraerse de dos formas según los planteamientos propuestos:

* Nivel estructural: se fija en tokens comunes y su posición para definir similitud entre frases. Este método presenta como ventaja una simplicidad de ejecución y un elevado rendimiento en contextos con heterogeneidad media (si es muy elevada no se agruparán; y si es muy baja serán todos el mismo), pero tiene limitaciones en cuanto a escalabilidad.

  Este método tiene dificultades para distinguir frases con estructura similar pero significados diferentes. Cuando el número de frases empieza a crecer, esta situación se vuelve insostenible.

  Por el contrario, su rendimiento es suficientemente bueno como para justificar su uso en determinadas situaciones en las que se cumplan ciertas condiciones.

* Nivel semántico: utiliza agrupaciones en función de la carga semántica. Este procesado es lento y tedioso, pero consigue crear distancias cortas para eventos con significados parecidos y más largas cuando el significado difiere en mayor medida. Esta cualidad, permite facilitar las tareas del predictor.

6.1.1 PROCESADO A NIVEL ESTRUCTURAL

Para el desarrollo a nivel estructural se ha planteado un sistema basado en múltiples autoencoders.

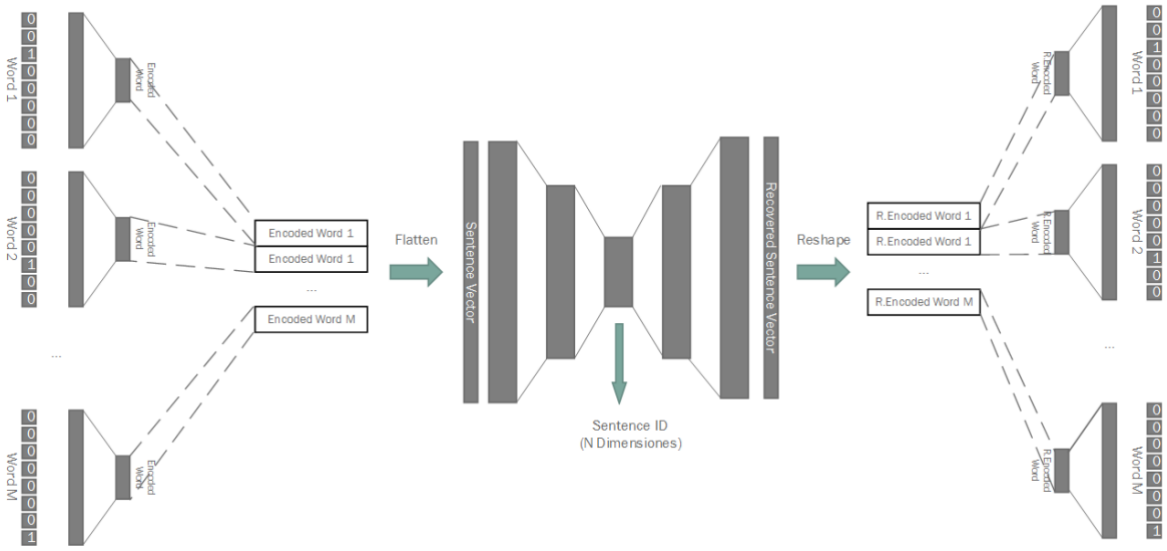

Figura 20 - Estructura del modelo de procesado de texto

En este contexto, se ha desarrollado un sistema doble de procesado de texto. En una primera capa se procesará cada palabra convirtiéndola en un vector de características. Posteriormente, en la segunda capa, se concatenarán los vectores de características de cada palabra de la frase para generar una representación numérica de la frase. Tras el paso de la segunda capa, esta representación numérica se reducirá a un número predefinido de elementos.

La *Figura 20* representa la estructura completa de procesado de descripciones.

6.1.1.1 Autoencoder de palabras

El entrenamiento del modelo dedicado a la codificación de las palabras requiere un preprocesado previo:

1. Se analiza la totalidad de los eventos para el problema descrito y se extraen todas las palabras utilizadas.
2. Se crea un diccionario de conversión texto a OneHot Encodding.

Una vez se dispone de todos los vectores OneHot que se deben codificar, se procede al entrenamiento de la red neuronal. Esta consistirá en un autoencoder que generará una constelación N-dimensional sobre el que la palabra quedará representada.

La Figura 21 y Figura 22 corresponde a un entrenamiento con salida resultante de dimensión 2. Se trata de una adaptación del modelo original para obtener una representación interpretable en una imagen En la red original hemos utilizado un vector de características de dimensión 30.

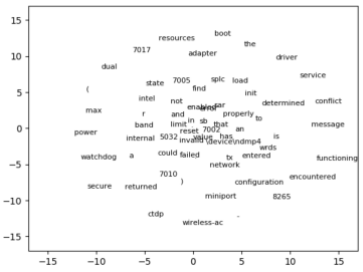

Figura 21 - Constelación de palabras

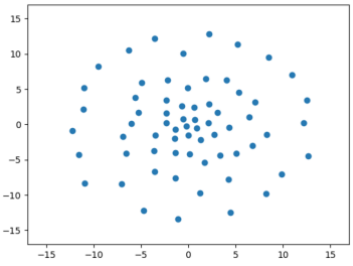

Figura 22 - Constelación con puntos

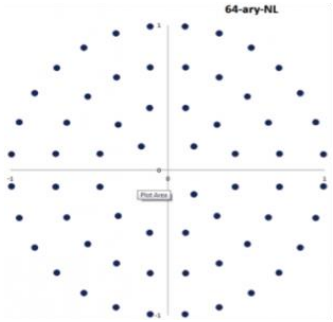

Figura 23 - Constelación 64-APSK

Al reducir la dimensión de salida a 2 para poder dibujarla en un plano, se ha obtenido una codificación de palabras tipo APSK. En este caso teníamos 66 palabras, por lo que compararemos la codificación con una codificación 64-APSK, que podemos ver en la Figura 23.

6.1.1.2 Autoencoder de frases

Una vez hemos codificado las palabras, realizamos la misma operación con las frases. Para ello realizamos el siguiente preprocesado:

1. Elegimos las frases que vamos a codificar (todas las existentes).
2. Realizamos una división en palabras
3. Codificamos las palabras individualmente
4. Formamos la matriz VxW formada por la concatenación de V palabras y W características de cada una.

Una vez tenemos la matriz que representa cada frase, procedemos a entrenar el algoritmo de autoencoding.

6.1.1.3 Problema con el tamaño de las frases

En la *Figura 24* vemos la constelación obtenida entrenando el algoritmo sin modificaciones. Como el algoritmo debe sostener un tamaño de mensaje fijo, este debe ser considerablemente elevado (para el desarrollo hemos elegido 60 palabras).

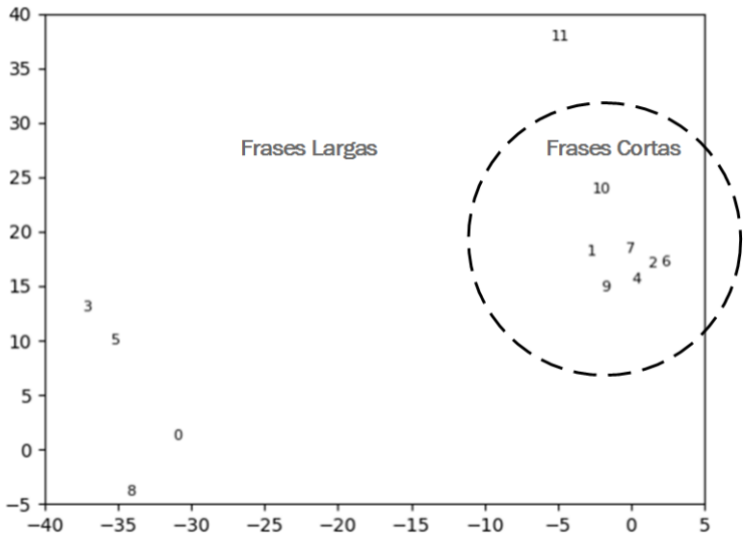

Figura 24 - Constelación con problema de sesgo. (Relación ID-Log en Tabla 4)

Este requisito genera un problema de sesgo en función del tamaño de la frase. Matemáticamente las redes neuronales suman diferentes pesos y esto no sería problema si la entrada de pesos fuera un número constante. Sin embargo, se está realizando un padding añadiendo ceros hasta completar el tamaño de entrada; lo que implica que una frase más larga contribuirá con un mayor número de pesos que otra.

Vemos dos grupos claramente diferenciados. El grupo de frases cortas, que son aquellas que hacen referencia a mensajes tipo (*'7010 - driver enabled (miniport init)'*); y la de frases largas con mensajes como (*'intel (r) dual band wireless-ac 8265 : has encountered a conflict in resources and could not load .'*,). A su vez, las frases largas se agrupan en el cluster 3-5-0-8, que incluye aquellas que empiezan por: *'intel (r) dual band wireless-ac 8265*.

Esta situación derivará en un problema en el entrenamiento del predictor. En función del nivel de agrupación de los diferentes ID, el predictor tendrá que adquirir un nivel mayor de precisión para coincidir con aquellos mensajes. Por tanto, será más probable que se realice una predicción sobre eventos separados que sobre eventos agrupados.

6.1.1.4 Solución al problema con el tamaño de las frases

Para solucionar este problema se ha realizado una modificación sobre el autoencoder, que implica hacer una corrección en función del tamaño de la frase. Esta mejora consiste en introducir una normalización de los valores tras pasar la primera capa; lo que corrige la dependencia de la salida de esta capa con el número de tokens diferentes de 0, es decir, que aportan valor. Con esta solución hemos obtenido la constelación de la *Figura 25*.

Pese a producirse una mejora, sigue habiendo eventos difíciles de identificar. Este sería el caso del cluster 2-7-11; cuyas descripciones podemos ver en la *Tabla 1*. Todas ellas tienen el mismo número de palabra y tienen una estructura similar en cuanto a la presencia de número, guion y paréntesis. Esta es la gran desventaja de este tipo de análisis.

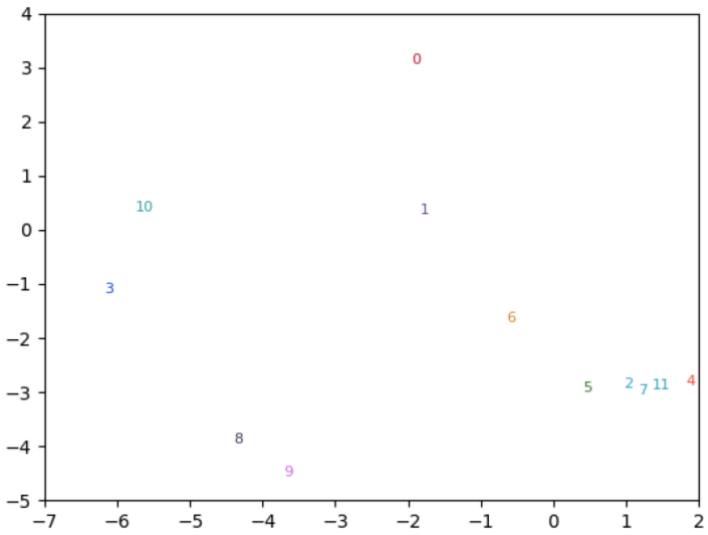

Figura 25 -Constelación con el sesgo por tamaño de frase corregido (Relación ID-Logs en la Tabla 5)

Tabla 1 - Relación ID - Log algunos valores de la Figura 25

Código de frase	***Frase***
2	'5007 - tx/cmd timeout (tfdqueue hanged)'
7	'7010 - driver enabled (miniport init)'
11	'7017 - secure boot (sb) configuration'

6.2 MODELO PREDICTIVO

Una vez se ha completado la identificación de eventos de manera numérica, procedemos a realizar la identificación de eventos.

En este punto es importante observar la distribución de eventos obtenida por las constelaciones. Con los métodos propuestos en el presente trabajo se obtiene una distribución basada en características no semánticas del texto. Esto supone un problema ya que dos eventos con características parecidas pueden tener significados opuestos y por tanto estar cerca en la constelación. Por este motivo es importante mantener una constelación en la que los eventos presenten distancias similares entre sus vecinos, es decir, estén uniformemente distribuidos. En caso contrario, será más probable que se predigan eventos cuanto más separados de los vecinos se encuentren.

Si la constelación se obtiene mediante un mecanismo más complejo, basado en la semántica, debería obtenerse una representación de un hiperplano en el que los eventos con un significado parecido deberían estar cercanos entre sí [12][14][16]. Por tanto, no es tan importante que estén uniformemente distribuidos. Si bien es cierto que los eventos con significado parecido (y por tanto más cercanos a sus vecinos) serán menos probables de predecir, podrá predecirse uno de la región cercana y por tanto con significado similar.

6.2.1 MODELO LSTM

El modelo LSTM mantiene el estado. Esta característica obliga a mantener N predictores si se tienen N flujos de información. Esto implica un gran uso de recursos a medida que se escala el sistema en el que se quiere implementar. Además, no se puede implementar un sistema de aprendizaje en streamming efectivo ya que no se comparten pesos y por tanto cada modelo aprendería sobre su flujo de información. No obstante, se trata de un modelo susceptible de ser implantado por su elevado nivel de madurez. Para cada escenario en función de los requisitos podrá implantarse un modelo u otro.

En la *Figura 26* se muestra la red LSTM elegida para este caso.

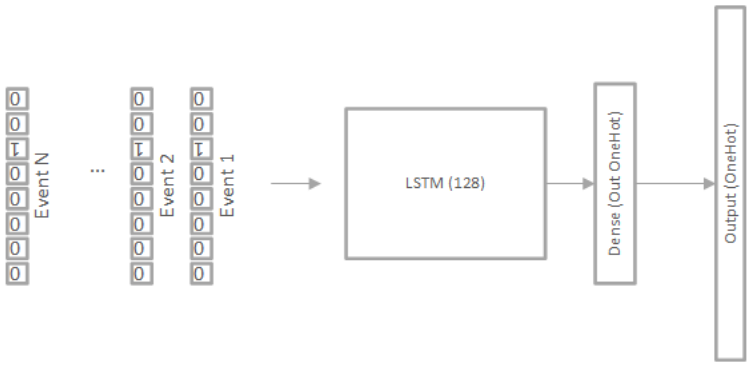

Figura 26 - Modelo LSTM

Solo se ha aplicado una capa de LSTM, proyectando las secuencias de eventos en un espacio de 128 dimensiones. A continuación, se combinan estos valores para generar la salida en la dimensión del OneHot Encodding.

Al tratarse de un modelo relativamente simple, su velocidad de entrenamiento es más elevada que Attention. Sin embargo, como se mencionó anteriormente, presenta ciertos inconvenientes, respecto al otro modelo.

6.2.2 MODELO ATTENTION

El modelo Attention es un modelo en que no se mantiene el estado, por este motivo la entrada deberá contener la secuencia completa de eventos sobre los que se extraerá la información. Nos basaremos en el modelo propuesto en el artículo "Attention Is All You Need", sobre el que realizaremos modificaciones para adaptar para nuestro caso de uso.

6.2.2.1 Análisis del modelo existente

La red original está pensada para la traducción de frases entre idiomas. Esto se debe tener en cuenta a la hora de analizar el modelo. En la *Figura 27* podemos encontrar un esquema de la red.

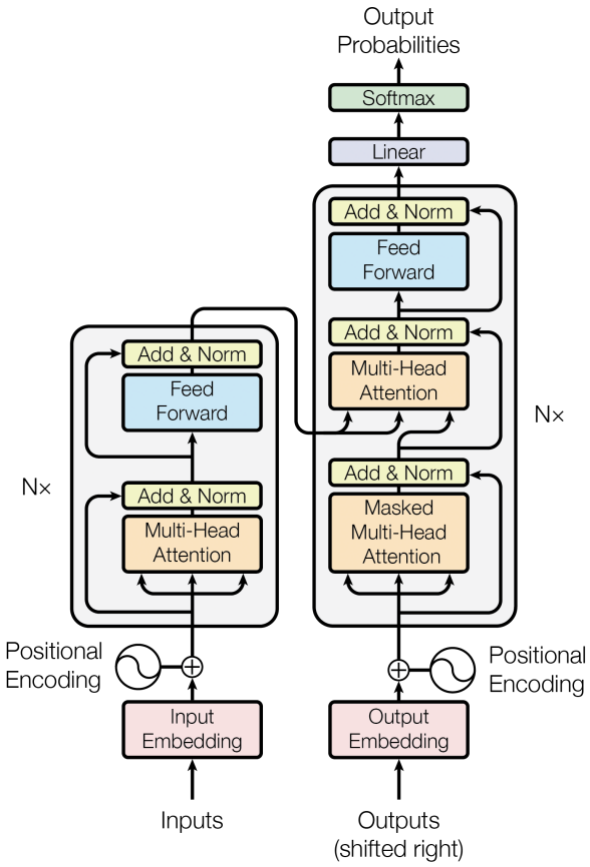

Figura 27 - Esquema de la red Transformer

6.2.2.2 Positional Encodding

En el modelo existente, la entrada consiste en una consecución de eventos en formato OneHot. Sobre esta entrada se realiza un Positional Encodding, que consiste en añadir una señal no periódica a la codificación OneHot, de manera que la red detecte mejor la secuencia de eventos.

La función propuesta para esta codificación es la ecuación E. 1. La incógnita *pos* hace referencia a la posición del evento dentro de la secuencia, i hace referencia a cantidad de eventos, y d_{model} a la dimensionalidad de cada evento.

$$E.\ 1 -> PE(pos, 2i) = sin(\ pos/10000^{2i/d_{model}})$$

Decidimos eliminar el Positional Encodding ya que no hemos comprobado que añada mejoras sustanciales. El uso de Positional Encodding hace necesario una entrada en tipo OneHot ya que sobre una codificación tipo constelación la amplitud es un propio factor de la codificación. Modificar puede crear un efecto aliasing entre los eventos. Por ejemplo, un evento 5.5 en una posición con PosEncodding +1 es equivalente a un evento +7 con una posición donde el valor del PosEncodding sea -0.5

6.2.2.3 Encoder

La siguiente capa que nos encontramos es lo que se denomina capa de encoder. En el modelo original se concatenan múltiples capas, pero por simplicidad reducimos el modelo a una única capa de encoding ya que el diccionario de eventos es mucho más reducido que el diccionario de palabras. El número de capas de encoder se convertirá en un parámetro de tunning del modelo.

El encodding consta de dos capas principales. La primera capa divide el evento en tres flujos de información denominados queries, keys y values. La *Figura 28* muestra el flujo de información de la primera capa. Las queries se pueden interpretar como un cambio de representación del evento a otro más oportuno. Los values son la aportación del evento codificado. Y los keys, enmascaran la query para que el evento representado por esta acceda a los valores deseado de los values.

Para guiar este comportamiento, es importante que la función de activación se encuentre antes de la creación de los values. De esta manera la máscara funcionará indicando que values debe leer, siempre entre 0 y 1; y los values definirán el valor de salida. Otra de las modificaciones introducidas ha sido cambiar la función softmax por tanh ya que, tras varios ensayos, había eventos que tendían a negativizar el valor de las keys. Esto se puede interpretar como eventos que indican que algo no se va a producir (por ejemplo, que indiquen que no va a ir bien).

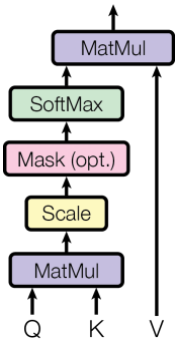

Figura 28 - Esquema de la Fase 1

La salida de esta fase debe tener la misma dimensionalidad que los eventos originales de manera que se pueda aplicar una suma del evento original con el resultado obtenido de esta primera fase. Con esto se pretende que se conozca el evento original y sea la red la que decida si lo modifica ligeramente o si lo transforma en un evento nuevo. Como los eventos originales se encontrarán normalizados, su entrada estará limitada entre 0 y 1. Si los valores generados a la salida del bloque Attention tienen valores pequeños, se estarán haciendo modificaciones sobre el evento; por el contrario, si toman valores con magnitudes superiores, se estará produciendo una recodificación completa del mismo (se considera despreciable el evento frente a los Values).

Tras esta fase, es necesario normalizar nuevamente los datos para evitar eventos con magnitudes muy dispares.

La segunda fase del codificador consiste en una red Feed-Forward convencional. Esta capa toma importancia cuando se concatenan múltiples encoders ya que desacoplan, dotando de libertad, las salidas de un encoder respecto a la entrada del siguiente.

6.2.2.4 Decoder

Tras las capas de encodding, se suceden capas de decoding. En el caso de la red original, destinada a traducción, se encarga de mezclar la entrada del idioma original con la parte de la frase ya traducida. De esta manera, se busca no solo que el significado cuadre en significado con la frase original, sino que, además se elija una palabra que encaje en el contexto del nuevo idioma (por ejemplo, frases hechas).

En el caso de los eventos, se busca dotar de cierta recurrencia a la red. Se busca cubrir la siguiente casuística: una secuencia predice que se pueden producir A o B, la siguiente secuencia, B o C y la siguiente B o D. La red puede predecir independientemente cualquiera de las dos posibilidades para cada caso; sin embargo, parece lógico que la predicción B sea tenida en cuenta ya que se mantiene presente en el tiempo. Para detectar este tipo de casuísticas es necesaria cierta realimentación.

En este caso; la realimentación también constituye una entrada, por lo que se puede mantener la filosofía de tener un solo modelo para los diferentes flujos de eventos. Solo es necesario mantener un buffer más que almacene las predicciones previas.

6.2.2.5 Capa de activación final

Por último, se añade una capa de codificación lineal que permite un ajuste final sobre la salida. Esta capa se encarga de combinar la salida del decodificador para ajustarla al tamaño de salida.

6.2.2.6 Transición Encoder - Decoder. La clave de Attention.

Una de las características claves de este modelo es la transición de encoder a decoder. En el modelo original, la salida del encoder indica cuales son las palabras a las que hay que prestar atención. Este conjunto de palabras almacena un concepto o conceptos que tendrán su imagen en el nuevo idioma.

Por tanto, debemos entender el encoder como un extractor de conceptos a un espacio dimensional genérico. Mientas que, el decoder, se encargará de transformar este espacio genérico en la representación del nuevo idioma. Esto se consigue en dos pasos:

* En la fase de encoding existen queries y keys, que centran la atención en ciertas palabras claves. Sobre estas, se realiza una multiplicación, que consiste en una aplicación de una máscara sobre los values. Estos values contienen la conversión del idioma original al espacio genérico.

* En la fase de decoding hay que destacar dos partes principales. Una primera etapa que actúa sobre la parte ya traducida, que genera los values del nuevo idioma. Y una segunda etapa en la que las queries y keys provienen de la salida del encoder (dimensión genérica) y los values de la etapa anterior en el nuevo idioma.

Este cambio de values de referencia es clave en escenarios de traducción. Sin embargo, en escenarios de predicción, donde el espacio de entrada y salida debería ser el mismo, no está claro que aporte mejoras. Por tanto, es un campo para explorar ya que, por una parte, puede ser interesante extraer información de predicciones anteriores; pero por otra, estas predicciones pueden aumentar la incertidumbre.

En los experimentos realizados, no se ha detectado mejoría o empeoramiento con la existencia o no de decoders; pero si un mayor tiempo de convergencia con decoders. El número de capas de encoders y decoders se considerará un hiperparámetro de la red.

6.2.2.7 Modelo adaptado

El modelo final desarrollado con las modificaciones propuestas se encuentra en la *Figura 29.*

Se ha llegado a este modelo mediante la prueba de diferentes orientaciones a la hora de entrenar la red [3][7][19]. A la vez que se monitorizaban los resultados con el visor Tensorboard.

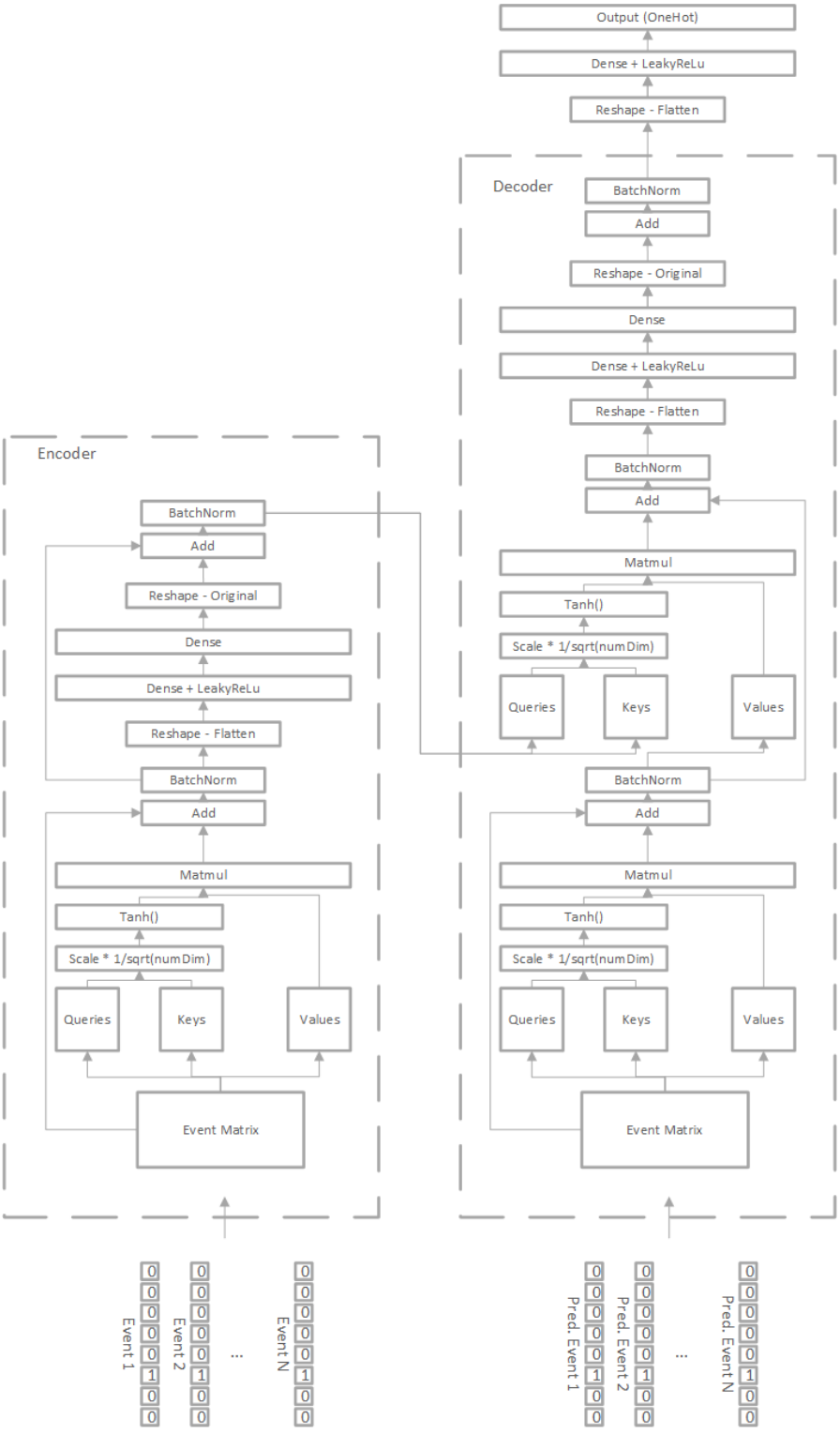

Figura 29 - Modelo Attention Adaptado

6.2.3 COMPARATIVA DE LOS MODELOS

Una vez descritos ambos modelos, procedemos a comparar las fortalezas de cada uno de ellos.

El modelo LSTM presenta como ventajas y desventajas:

- ✓ Tiempo de convergencia
- ✓ Precisión en corto tiempo
- ✓ Simplicidad del modelo
- × Necesidad de replicar para cada flujo de información
- × No aprendizaje en streaming (común)

En cuanto al modelo Attention:

- ✓ Mayor interpretabilidad
- ✓ Aprendizaje en Streaming
- ✓ Una sola réplica para todos los flujos
- × Tiempo de convergencia
- × Menor precisión que LSTM a largo plazo

La tabla de valores obtenida es la siguiente:

Tabla 2 - MAEs de la comparativa de modelos

	MAE (Mean Absolute Error)			
	10 Epoch	100 Epoch	400 Epoch	1000 Epoch
LSTM	0,0321	0,021	0,02	
1Enc. + 1Dec	0,075	0,053	0,042	0,036

Tabla 3 -MSEs de la comparativa de modelos

	MSE (Mean Square Error)			
	10 Epoch	100 Epoch	400 Epoch	1000 Epoch
LSTM	0,0233	0,0182	0,0185	
1Enc. + 1Dec	0,039	0,03	0,028	0,028

Todos estos valores de las *Tabla 2* y *Tabla 3* están en training ya que se busca un modelo capaz de aprender patrones con pocas repeticiones. El modelo está orientado a un aprendizaje en streaming de singularidades (como pueden ser ataques). Por tanto, nos encontramos en una situación no limitada por el overfitting; sino que cuanto más aprenda mejor. Además, el número de muestras de cada singularidad es reducido, por lo que eliminar una de estas muestras supone eliminar una parte importante de la información.

Un ejemplo de este caso sería el siguiente: Existe un evento que ha ocurrido un número reducido de veces (A+B->C). Crear un set de validación supondría eliminar muchas muestras en cuanto al total. Se pretende que el modelo sea capaz de inferir que, si vuelve a ocurrir A y B, existen probabilidades de que ocurra C.

Si nos ponemos en un caso más extremo, donde lo general es que A+B-> D, pero ocurren eventos como el del párrafo anterior; queremos que empiece a predecir o D y C, pero que C esté presente ya que puede ser importante.

Es decir, buscamos un resultado en el que la red informe por le parecido a situaciones y no por pura probabilidad. Si un evento A+B+N+M -> C y A+B+N+P -> D; y ocurre A+B+N+P queremos que la red prediga D con gran probabilidad, pero C esté presente puesto que cubre el 75% de los precedentes.

6.3 INTEGRACIÓN CON DEVOINC

Tras la definición de todos los bloques de manera individual, es necesario construir un pipeline que permita el flujo de datos entre Devo y la aplicación.

6.3.1 PROCESAMIENTO EN STREAMING

Una de las características críticas, definidas al inicio del proyecto, era la capacidad de procesado de datos en tiempo real. Para ello, hay que tener muy en cuenta el modelo utilizado (LSTM o Attention).

Si se opta por un modelo LSTM, la propia red neuronal se encargará de mantener el estado y procesar cada evento teniendo en cuenta la información de los anteriores. Esto hace necesario disponer de tantos predictores como secuencias de eventos existan (definidas en 5.3.2).

Por otra parte, si se elige el modelo de Attention, debe crearse un sistema de buffer que mantenga el estado; puesto que la entrada consiste en la secuencia completa de eventos. La ventaja de este método es que puede mantenerse un modelo único ya que es stateless.

En la *Figura 30* podemos ver el flujo de datos realizado para generar una predicción. Existe un proceso enganchado a la API de Devo, que se encarga de recibir el flujo de datos a medida que se reciben en la plataforma. Posteriormente este evento es inyectado en el predictor, que tendrá una estructura en función del modelo utilizado, y se genera una predicción de evento. A continuación, se prepara un evento en formato syslog y se envía a Devo para permitir el uso desde el API y por tanto compatibilizarlo con los framework ya existentes.

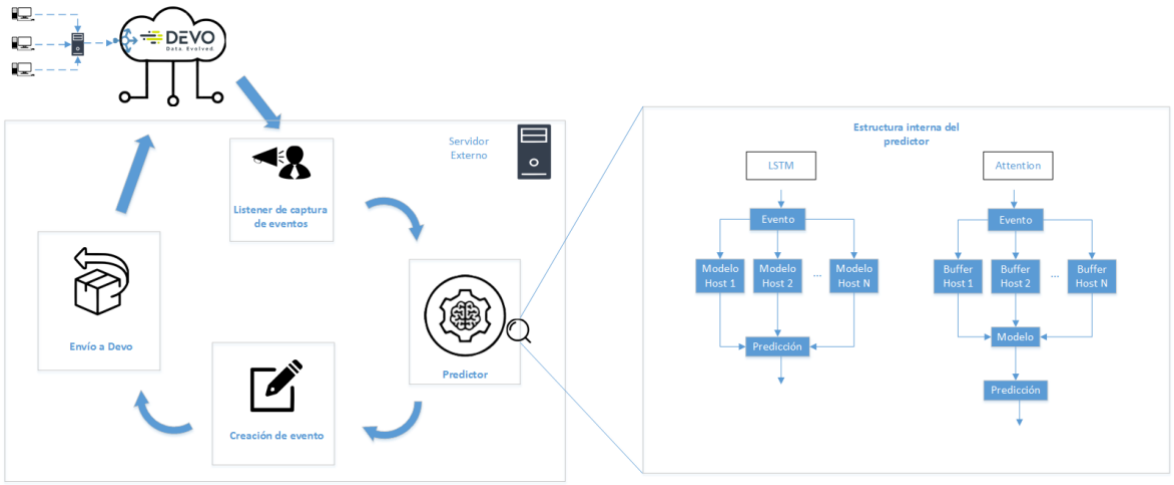

Figura 30 - Flujo de datos para el procesamiento en streaming

6.3.2 INTEGRACIÓN CON ALERTAS

La integración con las alertas de la plataforma se produce de manera automática gracias a la arquitectura planteada anteriormente. Una vez se realiza la predicción, esta se inyecta en una tabla nueva que hace referencia a las predicciones.

Al configurarse la predicción como un dato nuevo para la plataforma, se pueden usar todos los framework ya existentes. Entre otros; el aviso de alertas, que consiste en un listener enganchado a una determinada tabla que genera un aviso cuando se producen ciertas situaciones. Basta con configurar una alerta en la tabla de predicciones para generar alertas sobre la plataforma

6.3.3 DIAGRAMA RESUMEN DE COMBINACIONES DEL PROYECTO

En el campo de las redes neuronales, no siempre hay una solución que domine a las demás, sino que, dependiendo de la situación, unas tendrán un mayor rendimiento que otro. En la *Figura 31* mostramos un diagrama con las posibles rutas a seguir en el flujo de datos.

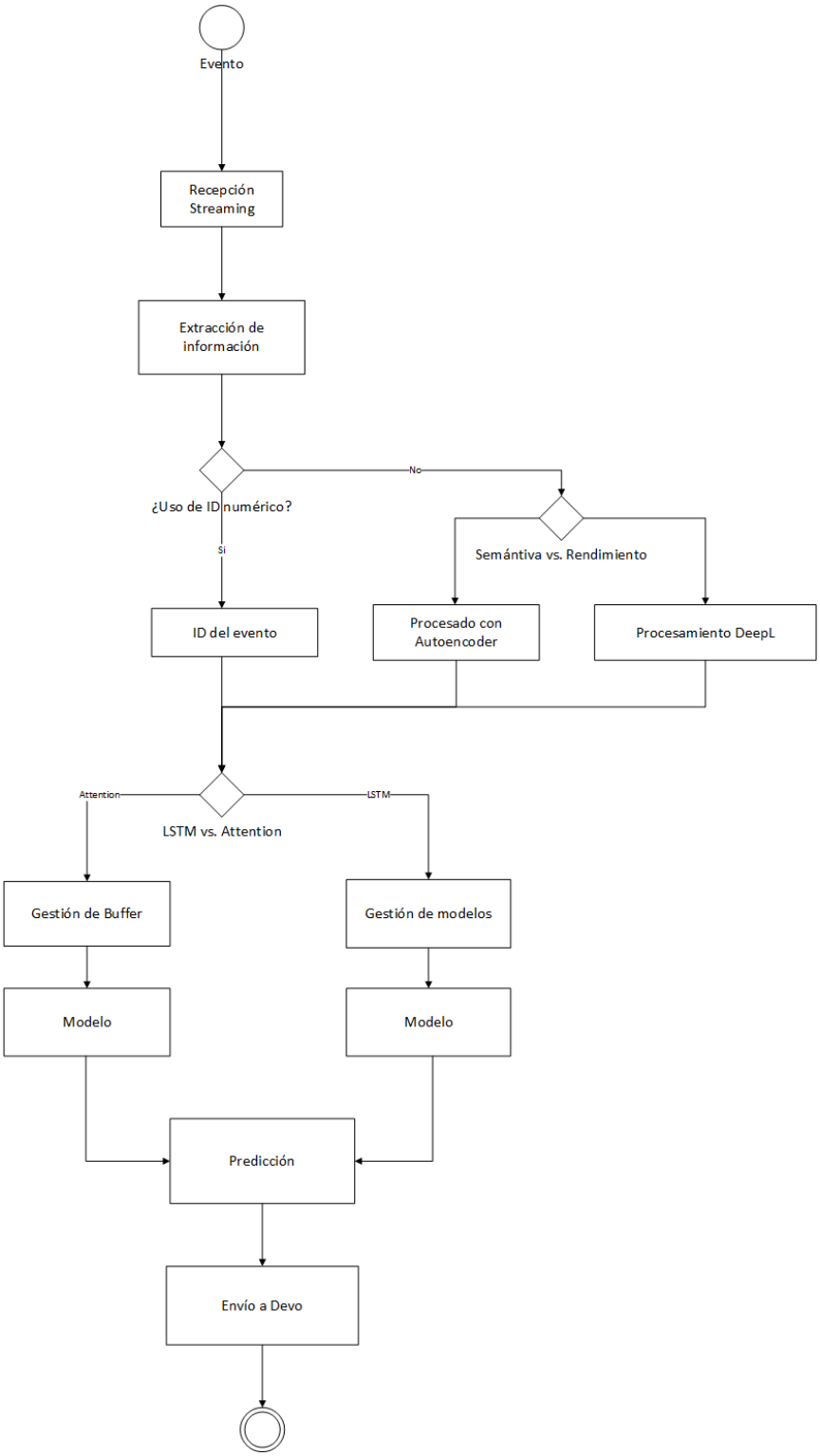

Figura 31 - Diagrama resumen del predictor

Capítulo 7. CICLO DE MEJORA I: VISUALIZACIÓN DE PREDICCIONES

Para visualizar como actúa el predictor frente a la secuencia real de eventos, se ha diseñado la visualización de la *Figura 32*. En ella se observa un diagrama de burbujas. El eje X corresponde con la secuencia temporal, mientras que el Y contiene los diferentes valores que se pueden obtener. Posteriormente, se dibujan círculos en las posiciones del entramado donde se ha establecido una probabilidad de que ocurra cierto evento. El tamaño de las burbujas indica como de probable es que ocurra dicho evento.

Cabe destacar que el tamaño se define según el área, y no el radio; por lo que un valor doble que otro se verá reflejado con el doble de área, pero un pequeño incremento en el radio. Además, se ha establecido un valor superior para los eventos reales con el fin de eliminar superposiciones; por lo que, aun alcanzando el máximo valor en la predicción, no se alcanzará el tamaño del evento real.

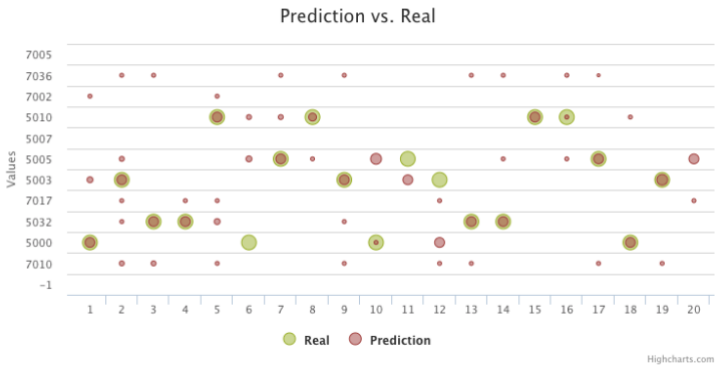

Figura 32 - Visualización de una secuencia de eventos sincronizada con sus predicciones

Capítulo 8. CICLO DE MEJORA II: PREDICCIÓN Y APRENDIZAJE EN STREAMING

Para la realización del procesado en streaming se ha planteado la arquitectura de los *Figura 33*. Esta arquitectura está inspirada en las arquitecturas Lambda. Se ha planteado por una parte un sistema de aprendizaje, que recibe los eventos y actualiza progresivamente el modelo en mini-batches. Por otra parte, existe un modelo, que se actualiza al finalizar cada mini-batch, que hace predicciones en streaming.

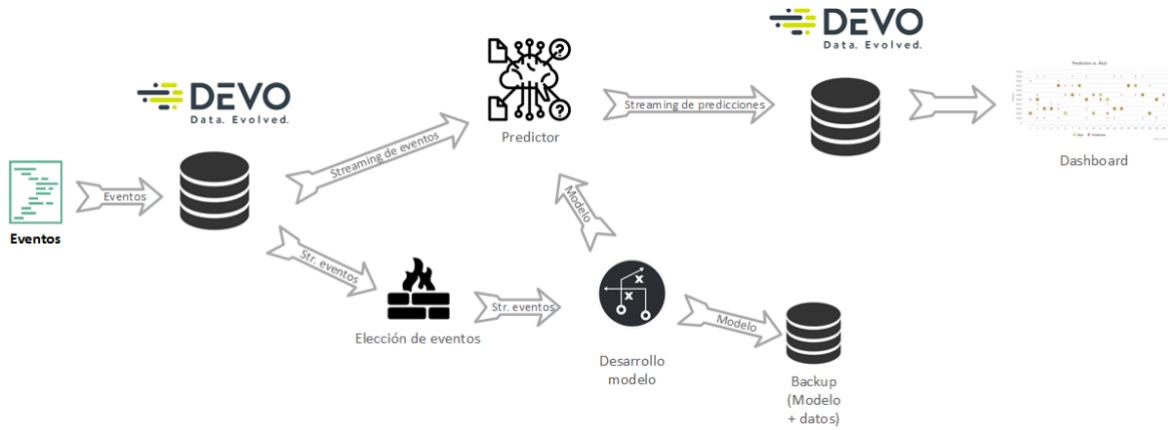

Figura 33 - Arquitectura de procesamiento streaming

El flujo de información consiste en una recepción continua de eventos desde la plataforma Devo. El sistema de predicción se enganchará mediante el framework streaming de la plataforma. En este punto el flujo de datos se divide en dos: por una parte, se realizan las predicciones oportunas de los eventos, y por otra se redirige a un selector de eventos para el entrenamiento del modelo.

En la rama del predictor se procesan eventos continuamente con un modelo dado. Este modelo debe estar desconectado del modelo de entrenamiento para garantizar el funcionamiento de este. El modelo de entrenamiento es susceptible de sufrir modificaciones

estructurales, no solo de los pesos de la función; por lo que es necesario mantener independencia entre los mismos. Una vez se ha realizado la predicción, se reconstruirá el evento (ID, frase asociada, severity, etc.) y se enviará a la plataforma Devo. En este punto, se encadena la información con el sistema explicado en el *Capítulo 7.* para permitir la visualización de los eventos (su utilidad depende de la cantidad de eventos/segundo producidos). Además, queda integrado con el sistema de alertas descrito en el *6.3.2*; por lo que no hace falta una supervisión del sistema, sino una simple configuración de alertas.

En la otra rama, se encuentra la creación de modelos. El flujo de eventos llega a un sistema de filtrado. Una vez se ha decidido cuales son los eventos importantes; y por tanto aquellos que aprenderá el modelo, se cargan en el dataset para que pasen a ser procesados por el sistema de entrenamiento.

8.1 CONSIDERACIONES DEL MODELO.

Para la implantación del sistema de procesamiento en streaming se han realizado una serie de modificaciones en el modelo propuesto en el apartado *6.2.2*.

8.1.1 SELECCIONADOR DE EVENTOS

El sistema original se basaba en un modelo basado en overfitting para la predicción. Esto hace que el sistema sea capaz de generar alertas con un bajo numero de repeticiones (como puede ser el caso de nuevos ataques). Es decir, se quiere que el sistema alerte de un posible ataque la segunda o tercera vez que ocurra; de manera que se pueda promover una actuación para investigar si se trata o no de un riesgo real.

Esta característica hace que sea especialmente importante definir que eventos serán susceptibles de pasar a la fase de entrenamiento. Por ello se ha desarrollado un paso anterior a la incorporación de los eventos al dataset de entrenamiento; que consiste en un filtro basado en la intervención humana o mediante programación. Los eventos que pasen este filtro

pasarán al sistema de aprendizaje, donde el modelo aprenderá todo lo que pueda dicho evento.

8.1.2 ONEHOT VS. CONSTELACIÓN

En la literatura aparecen dos grandes tendencias a la hora de trabajar con redes neuronales:

* OneHot Encodding: consiste en la existencia de una cifra para cada categoría (tipo de evento en este caso). Esta codificación tiene como ventaja que esta salida indica como de probable es que ocurra cierto evento. Por el contrario, no permite un aumento del numero de ocurrencias ya que esto supondría hacer modificaciones estructurales.
* Codificación tipo constelación. Se basa en la codificación de un número de eventos para un número determinado de entradas. Permite flexibilidad en cuanto al número de eventos, ya que solo hay que encontrar una nueva combinación de las cifras para representar el nuevo evento. Pero tiene dos desventajas importantes; por una parte, la colocación de los eventos según su parecido real es importante a nivel de convergencia del modelo (tiempo y/o precisión). Por otra, no se conoce como de probable es un evento.

  Pueden realizarse inferencias como la siguiente: si un evento espera la salida 1 y otro la salida 2, y el predictor predice 1.7; se puede decir que es un 70% probable que esté en el evento 2 y un 30% de que esté en el evento uno. Sin embargo, estas aproximaciones son peligrosas ya que no necesariamente una aproximación lineal es correcta.

  Además, si la colocación de eventos en la constelación no es adecuada, pueden darse casos en los que dos eventos muy relacionados sean muy lejanos; y por tanto el punto inferido sea un punto intermedio entre ellos. Al estar alejados, es probable que este punto intermedio interfiera con un tercer evento que diste en la realidad de los mismo.

Se ha elegido el formato OneHot Encodding. Además, el uso de esta codificación presenta ventajas en el modelo de Attention.

8.1.3 CRECIMIENTO DEL MODELO

Como el sistema de aprendizaje en streaming es susceptible de recibir eventos que no habían ocurrido antes, se tiene que corregir el problema generado por la codificación OneHot. Esto se consigue haciendo crecer la estructura del modelo para que reciba un mayor número de entradas y de salidas.

Tensorflow no está diseñado para soportar modelos completamente flexibles, por lo que el modelo crece, se almacena y se reconstruye cada vez que se incorporan nuevos tipos de eventos [6][11][17]. Paralelamente es necesario hacer crecer los historiales almacenados y el set de entrenamiento.

El siguiente fragmento de código permite hacer crecer el modelo:

```
self.oneFeatureExpand.append({
                'original': V,
                'sliceVector': [[0, 0], [1, -1]],
                'tileVector': [-1, 1],
                'appendAxis': 0
            })

def increaseOneHot(self, nTimes, saver, out_dir):
        i=0
        prev_op=None
        for op in self.oneFeatureExpand:
            oneSlice = tf.slice(op['original'], op['sliceVector'][0],
op['sliceVector'][1])
            tileVector=[nTimes if x ==-1 else x for x in op['tileVector']]
            concat = tf.concat([op['original'],tf.tile(oneSlice,tileVector)],
op['appendAxis'])
            self.sess.run(tf.assign(op['original'], concat,
validate_shape=False))
            #temp = self.sess.run(op['original'])

        saver.save(self.sess, os.path.join(out_dir,
'model_{0:05d}.ckpt'.format(i)))
```

En primer lugar, se define detrás de cada capa cual es el procedimiento para hacer crecer dicha capa. Debe especificarse: capa a crecer, forma de obtener un crecimiento de 1, dirección de crecimiento y forma de unión.

Estos datos servirán a la función increaseOneHot para definir la forma de incremento de la estructura del modelo. La *Figura 34* muestra un esquema del proceso para el crecimiento del modelo.

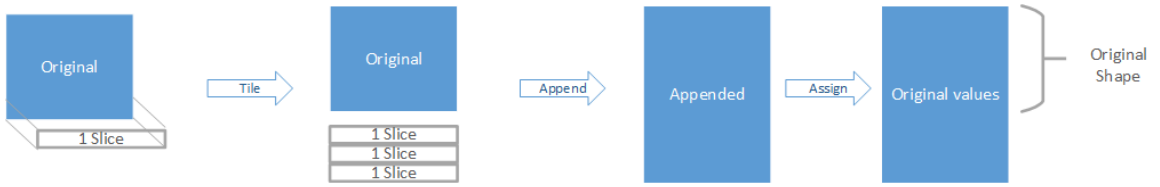

Figura 34 - Crecimiento de una capa

Tensorflow no permite crear nuevos nodos con un mismo nombre ni modificar los ya existentes, por lo que crear un nuevo nodo unificado no serviría para crear un modelo capaz de crecer una segunda vez ya que, al guardarlo y restaurarlo, se mantendría el nuevo nombre; y por tanto habría que cambiar el nombre continuamente con cada incremento. Una de las características clave de este proceso es la asignación sin validación. Esto permite saltarse comprobaciones y asignar tensores a nombres con otras características.

Una vez se ha realizado la asignación; no se pueden seguir realizando operaciones de entrenamiento ya que los valores de los pesos no encajan en forma con los asignados a los tensores. Sin embargo, se puede guardar el nuevo modelo; y este se guardará con los nuevos pesos asignados.

Si restauramos grafo y pesos, no habremos solucionado el problema; pero si creamos un nuevo modelo (manteniendo nombres) y sobre el restauramos los pesos; tendremos un nuevo gráfico en el que los valores si se adaptan las formas con el contenido de las variables y podremos continuar con el aprendizaje.

8.1.4 APRENDIZAJE TIPO GAN

Uno de los problemas del modelo se encuentra en la existencia de capas Feed-Forward al final de cada etapa. Estas capas permiten combinar las transformaciones realizadas sobre los diferentes eventos para generar un solo evento. No obstante, al estar después de cada capa,

se encuentra antes en las iteraciones del back-propagation. Esto hace que se tienda a aprender características en estas capas antes que por la transformación de las Queries, Keys y Values. Lo que hace que el tiempo de convergencia crezca y que se produzca cierta inestabilidad. Para evitar esto, se ha planteado un sistema de aprendizaje inspirado en las redes GAN. Por un lado, se entrenará el combinador de características y por otro el transformador de características.

8.2 DIAGRAMA DE SECUENCIA DEL PROGRAMA COMPLETO

En la *Figura 35* podemos ver un diagrama de actividad del programa completo.

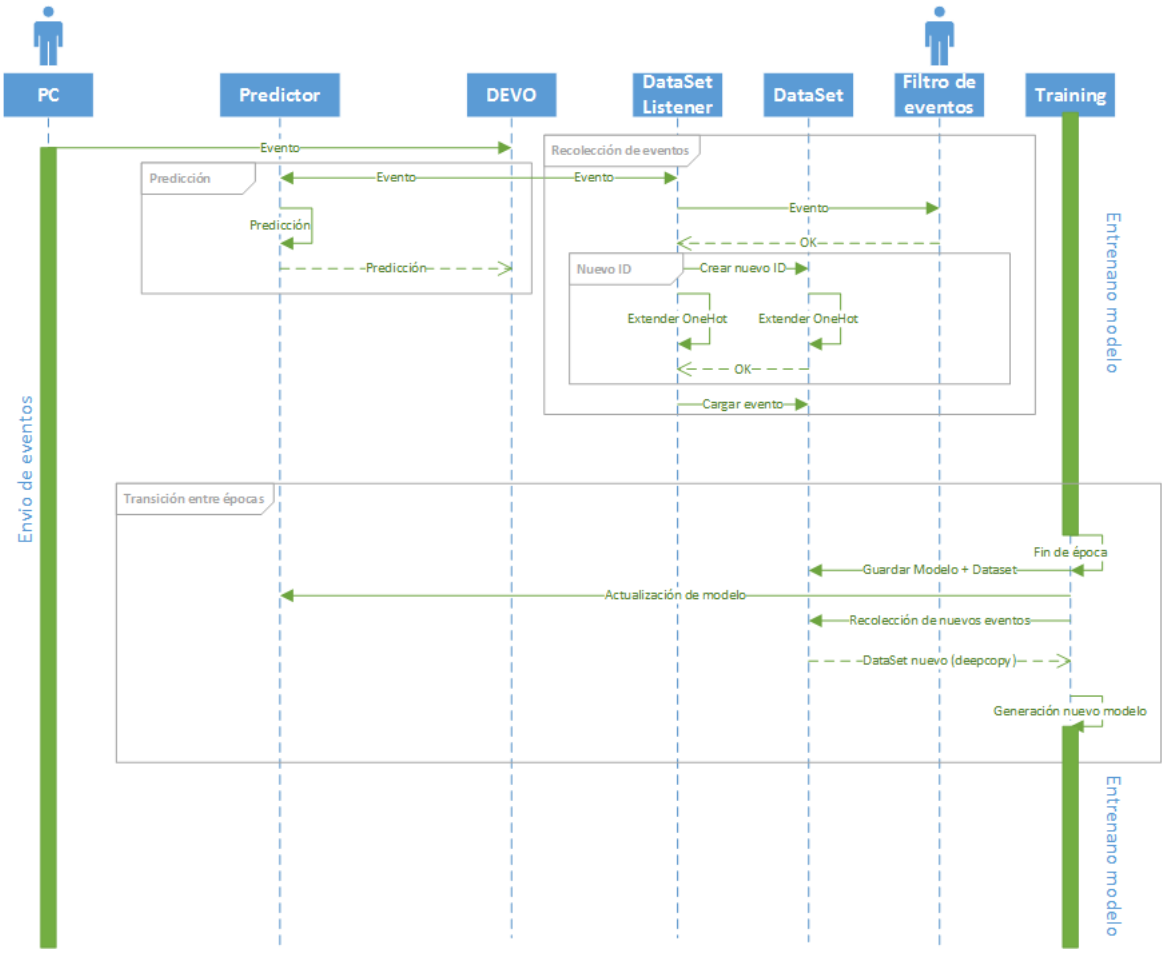

Figura 35 - Diagrama de secuencia del programa completo

En el diagrama, se muestran las acciones realizadas por las tareas de recepción de eventos, predicción, recolección de nuevos eventos y entrenamiento. Todas estas tareas ocurren de manera paralela. Es necesario hacer una gestión de los recursos para evitar problemas de concurrencia.

Las diferentes tareas se han desacoplado con las siguientes medidas:

* El modelo de predicción y de training es diferente. Cada ve que acaba un entrenamiento el modelo se almacena y se carga de nuevo.
* El training hace un snapshot del dataset cada vez que comienza una época. De esta manera garantiza que no va a necesitar modificaciones estructurales a mitad de la época, provocadas por la llegada de un evento no contemplado en la codificación OneHot.

Con estas medidas, se consigue un procesamiento en forma de mini-batch con el que los procesos pueden ejecutarse de forma asíncrona. El tamaño del mini-batch viene definido por el número de iteraciones que se realizan en el modelo en una época.

Este parámetro suele coincidir con aquel que recorra todas las muestras del dataset una vez, pero como hemos aplicado bootstrap con compensación de clases para corregir el sesgo del datases; el tamaño del mini-batch queda a elección del usuario. Cuanto más elevado sea el parámetro, mayor tiempo de entrenamiento respecto a las tarareas de transición; pero mayor será el tiempo de respuesta ante eventos nuevos.

En el desarrollo de las pruebas hemos utilizado un número de iteraciones para que se actualice el modelo cada 1 minuto aproximadamente.

Capítulo 9. CONCLUSIONES Y TRABAJOS FUTUROS

El estado actual del proyecto presenta las siguientes opciones para continuar el desarrollo:

* Construir un modelo para detectar nuevas situaciones. El sistema proporciona un framework genérico para la inferencia de patrones de comportamiento. Si en lugar de introducir una secuencia de eventos individuales, se introduce una secuencia de situaciones (agrupaciones de eventos); se puede comenzar a inferir a más alto nivel. Por ejemplo, si A+B generan C, nuestro modelo se aprendería dicha relación. Sin embargo, concatenando el modelo podríamos detectar que se ha dado la situación A+B->C junto con otra situación D+E->F. La existencia de ambas situaciones (C y F) puede indicar la sucesión de un nuevo evento.

  Esta misma funcionalidad puede conseguirse aumentando el histórico temporal de eventos; pero los resultados de agrupar pueden no ser los mismos que aumentar el histórico temporal.
* Aumentar el límite temporal. En combinación con la propuesta anterior, se puede analizar el uso del modelo para un avance temporal mayor (como pueden ser horas). Se pueden poner en paralelo modelos entrenados con una diferencia temporal, de manera que cada uno esté centrado, por ejemplo, en 1 hora, 2 horas y 3 horas respectivamente. Si un evento va a ocurrir tendrá que pasar por los 3 a medida que se aproxima en el tiempo; y además según que predictor se active, podrá indicar cuanto tiempo queda para el evento.
* Mejorar la predicción. Se puede realizar un estudio de los diferentes hiperparámetros, así como la forma de entrenamiento
* Asignación de ID mediante procesado de texto. Desarrollar un sistema de análisis de texto que permita asignar el ID en función del texto producido, por ejemplo; que un evento “Shutdown in 30 seconds” sea del mismo tipo que uno “Shutdown in 10 seconds”.

* Cambio de modelo a LSTM. Desarrollo de un sistema que aproveche la mayor precisión de este modelo. Por ejemplo, que sea capaz de tener un modelo único para todas las secuencias temporales mediante la gestión de buffers de los valores de la memoria de las LSTM; de manera que pueda retomarse un estado correspondiente a otra secuencia cuando va a inyectarse el siguiente evento de esa secuencia.

Capítulo 10. BIBLIOGRAFÍA

[1] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gómez, A.N.; Kaiser, L.; Polosukhin, I.. "ATTENTION IS ALL YOU NEED". Google Brain, Google Research & University of Toronto. Diciembre, 2017: https://arxiv.org/abs/1706.03762

[2] Bai, S.; Kolter, J.Z..; Koltun, V. "AN EMPIRICAL EVALUATION OF GENERIC CONVOLUTIONAL AND RECURRENT NETWORKS FOR SEQUENCE MODELING". Abril, 2018: https://arxiv.org/pdf/1803.01271.pdf

[3] Popel, M.; Bojar, O.. "TRAINING TIPS FOR THE TRANSFORMER MODEL". Charles University. Mayo 2018: https://arxiv.org/pdf/1804.00247.pdf

[4] Alammar J. "THE ILLUSTRATED TRANSFORMER" Junio, 2018: http://jalammar.github.io/illustrated-transformer/

[5] Graves, A. "GENERATING SEQUENCES WITH RECURRENT NEURAL NETWORKS" University of Toronto. Junio 2014 https://arxiv.org/pdf/1308.0850.pdf

[6] Sachan, A. "A QUICK COMPLETE TUTORIAL TO SAVE AND RESTORE TENSORFLOW MODELS" https://cv-tricks.com/tensorflow-tutorial/save-restore-tensorflow-models-quick-complete-tutorial/

[7] Ruder, S. "AN OVERVIEW OF GRADIENT DESCENT OPTIMIZATION ALGORITHMS" Insight Centre for Data Analytics, NUI Galway. Septiembre 2016 https://arxiv.org/pdf/1609.04747.pdf

[8] Edward, C.; Taha Bahadori, M.; Schuetz, A.; Stewart, W.F.; Sun, J.. "DOCTOR AI: PREDICTING CLINICAL EVENTS VIA RECURRENT NEURAL NETWORKS" Septiembre 2016: https://arxiv.org/pdf/1511.05942.pdf

[9] Brownlee, J.. "HOW TO DEVELOP A BIDIRECTIONAL LSTM FOR SEQUENCE CLASSIFICATION IN PYTHON WITH KERAS" Junio 2017: https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/

[10] Brownlee, J.. "HOW TO USE THE TIMEDISTRIBUTED LAYER FOR LONG SHORT-TERM MEMORY NETWORKS IN PYTHON" Mayo 2017: https://machinelearningmastery.com/timedistributed-layer-for-long-short-term-memory-networks-in-python/

[11] Hajek, B.. "IMPORTING MULTIPLE TENSORFLOW MODELS (GRAPHS)" Abril 2017: https://bretahajek.com/2017/04/importing-multiple-tensorflow-models-graphs/

[12] Karani, D.. "INTRODUCTION TO WORD EMBEDDING AND WORD2VEC" Septiembre 2018: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

[13] Culurciello, E.. "NEURAL NETWORKS BUILDING BLOCKS" Noviembre 2017: https://medium.com/@culurciello/neural-networks-building-blocks-a5c47bcd7c8d

[14] Ahamed, S.. "TEXT CLASSICATION USING CNN, LSTM AND PRE-TRAINED GLOVE WORD EMBEDDINGS: PART-3" Enero 2018: https://medium.com/@sabber/classifying-yelp-review-comments-using-cnn-lstm-and-pre-trained-glove-word-embeddings-part-3-53fcea9a17fa

[15] Graves, A.; Mohamed, A.; Hinton, G.. "SPEECH RECOGNITION WITH DEEP RECURRENT NEURAL NETWORKS" University of Toronto. Marzo 2013 https://arxiv.org/pdf/1303.5778.pdf

[16] Polosukhin, I.. "TENSORFLOW—TEXT CLASSIFICATION" Noviembre 2016: https://machinelearnings.co/tensorflow-text-classification-615198df9231

[17] Morgan. "TENSORFLOW: SAVING/RESTORING AND MIXING MULTIPLE MODELS" Noviembre 2016: https://blog.metaflow.fr/tensorflow-saving-restoring-and-mixing-multiple-models-c4c94d5d7125

[18] Malik, U.. "TIME SERIES ANALYSIS WITH LSTM USING PYTHON'S KERAS LIBRARY" Noviembre 2018 https://stackabuse.com/time-series-analysis-with-lstm-using-pythons-keras-library/

[19] Kattan, A.; Fatima, S.; Arif, M.. "TIME-SERIES EVENT-BASED PREDICTION: AN UNSUPERVISED LEARNING FRAMEWORK BASED ON GENETIC PROGRAMMING" Abril 2015: https://doi.org/10.1016/j.ins.2014.12.054

[20] Olah, C.. "UNDERSTANDING LSTM NETWORKS" Agosto 2015: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Capítulo 11. BIBLIOGRAFÍA DE IMÁGENES

[1] Figura 1: Olah, C.. "UNDERSTANDING LSTM NETWORKS". Colah Github. Agosto, 2015: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[2] Figura 2: Olah, C.. "UNDERSTANDING LSTM NETWORKS". Colah Github. Agosto, 2015: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[3] Figura 3: Graves, A.; Mohamed, A.; Hinton, G.. "SPEECH RECOGNITION WITH DEEP RECURRENT NEURAL NETWORKS". Department of Computer Science, University of Toronto. Marzo, 2013: https://arxiv.org/abs/1303.5778

[4] Figura 4: Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gómez, A.N.; Kaiser, L.; Polosukhin, I.. "ATTENTION IS ALL YOU NEED". Google Brain, Google Research & University of Toronto. Diciembre, 2017: https://arxiv.org/abs/1706.03762

[5] Figura 5: Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gómez, A.N.; Kaiser, L.; Polosukhin, I.. "ATTENTION IS ALL YOU NEED". Google Brain, Google Research & University of Toronto. Diciembre, 2017: https://arxiv.org/abs/1706.03762

[6] Figura 6: Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gómez, A.N.; Kaiser, L.; Polosukhin, I.. "ATTENTION IS ALL YOU NEED". Google Brain, Google Research & University of Toronto. Diciembre, 2017: https://arxiv.org/abs/1706.03762

ANEXO A

En este anexo mostramos las tablas que asocian un número identificador a los diferentes logs de las figuras contenidas en el texto:

Tabla 4 - Relación de logs para la constelación sin corrección de sesgo (Figura 24)

Código de frase	***Frase***
0	'intel (r) dual band wireless-ac 8265 : could not find a network adapter .’
1	'message'
2	'7017 - secure boot (sb) configuration'
3	'intel (r) dual band wireless-ac 8265 : has encountered a conflict in resources and could not load .'
4	'7010 - driver enabled (miniport init)',
5	'intel (r) dual band wireless-ac 8265 : has encountered an internal error and has failed .'
6	'7005 - sar value max tx power (wrds)'
7	'5007 - tx/cmd timeout (tfdqueue hanged)'
8	'intel (r) dual band wireless-ac 8265 : the network adapter has returned an invalid value to the driver .'
9	'5032 - driver miniport reset watchdog'
10	'7002 - ctdp power limit value (splc)'
11	'the \\device\\ndmp4 service entered the intel (r) dual band wireless-ac 8265 state .'

Tabla 5 - Relación de logs para la imagen con sesgo corregido (X)

Código de frase	***Frase***
0	'the \\device\\ndmp4 service entered the intel (r) dual band wireless-ac 8265 state .'
1	'7002 - ctdp power limit value (splc)'
2	'5007 - tx/cmd timeout (tfdqueue hanged)'
3	'intel (r) dual band wireless-ac 8265 : could not find a network adapter .'
4	'7005 - sar value max tx power (wrds)'
5	'5032 - driver miniport reset watchdog'
6	'message'
7	'7010 - driver enabled (miniport init)'
8	'intel (r) dual band wireless-ac 8265 : has encountered an internal error and has failed .'
9	'intel (r) dual band wireless-ac 8265 : has encountered a conflict in resources and could not load .'
10	: 'intel (r) dual band wireless-ac 8265 : the network adapter has returned an invalid value to the driver .'
11	'7017 - secure boot (sb) configuration'